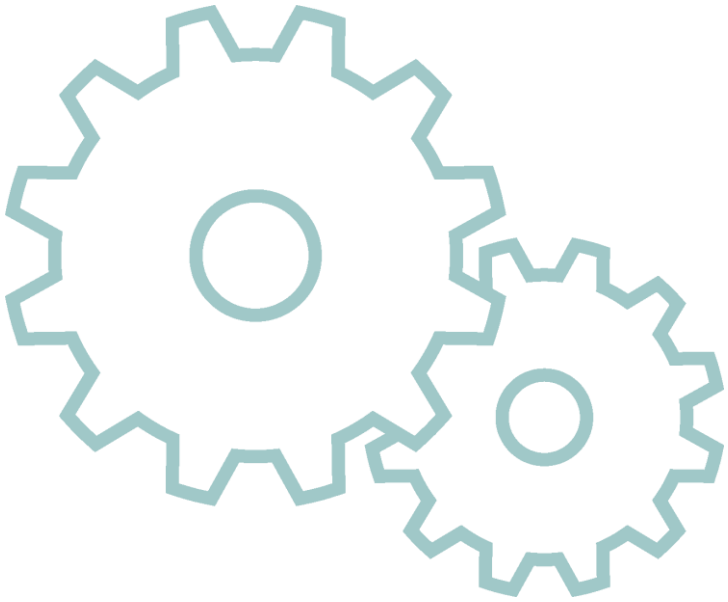
Bachelor-Arbeit im Studiengang Medientechnik/Wirtschaft[plus]

INTEROPERABILITÄT IM INTERNET DER DINGE

Konzeption und Implementierung unter Verwendung der CloudRail-API



CHRISTIAN SCHNEIDER

Erstbetreuer: Prof. Dr. Tom Rüdebusch
Zweitbetreuer: Florian Wendel (CloudRail)

Hochschule Offenburg
Medien und Informationswesen
Wintersemester 2015/2016

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API ..... *Application Programers Interface*
AWS ..... *Amazon Web Services*
HTTPs ..... *Hypertext Transfer Protocol Secure*
IaaS ..... *Infrastructure as a Service*
IDE ..... *Integrated Development Environment*
IoE ..... *Internet of Everything*
IoT ..... *Internet of Things*
IP ..... *Internet Protocol*
JRE ..... *Java Runtime Environment*
JSON ..... *JavaScript Object Notation*
LiDAR ..... *Laser illuminated Detection and Ranging*
M2M ..... *Machine to Machine*
OBD ..... *On Board Diagnose*
P2P ..... *Peer-to-Peer*
PaaS ..... *Platform as a Service*
REST ..... *REpresentional State Transfer*
RFID ..... *Radio Frequency Identification*
RWE ..... *Rheinisch-Westfälisches Elektrizitätswerk AG*
SaaS ..... *Software as a Service*
SASS ..... *Syntactically Awesome Stylesheets*
SDK ..... *Software Development Kit*
SD-Karte ..... *Secure Digital Memory Card*
SLA ..... *Service level agreement*
SOAP ..... *Simple Object Access Protocol*
SoC ..... *System on a Chip*
SSL/TLS ..... *Secure Sockets Layer/Transport Layer Security*
TCP ..... *Transmission Control Protocol*
UDP ..... *User Datagram Protocol*
UPnP ..... *Universal Plug and Play*
URI ..... *Uniform Resource Identifier*
URL ..... *Uniform Resource Locator*
USB ..... *Universal Serial Bus*
WAR ..... *Web Application Archive*
WinAPI ..... *Windows Application Programers Interface*
WLAN ..... *Wireless Local Area Network*
XML ..... *eXtensible Markup Language*
XML-RPC ..... *eXtensible Markup Language Remote Procedure Call*
YAML ..... *Yet another multicolumn layout*

Einleitung

Smart Home, APIs, Ubiquitous Computing, Cloud Computing – all diese Begriffe sind Teil eines großen Trends. Das Internet der Dinge (IoT) vernetzt, wie der Name es vermuten lässt, Dinge oder Alltagsgegenstände. Es entsteht eine Umgebung, in der sich Geräte gegenseitig ansprechen und steuern können. So geht der Rollladen automatisch hoch, wenn der Wecker klingelt oder der Briefkasten sendet eine Mitteilung an das Smartphone, sobald ein Brief eingegangen ist. Darüber hinaus steuern sich Geräte mit Hilfe von Sensoren und Informationen aus dem Netz selbst, wie es das lernende Raumthermostat von Nest zeigt. Es lernt die Gewohnheiten der Hausbewohner und stellt die Temperaturen nach Tages- und Nachtzeiten automatisch ein. Außerdem wird über Bewegungssensoren erkannt, wann die Eigentümer zu Hause sind. Ist das Haus leer, wird stromsparend die Heizung heruntergefahren. [18] Der Begriff Smart Home gewinnt zunehmend an Bedeutung. Er beschreibt ein automatisiertes Zuhause, das sich, wie beschrieben, selbst reguliert.

Aber nicht nur für Privatpersonen ist das IoT ein interessantes Werkzeug. Längst haben Firmen die Vorzüge einer automatisierten Umgebung erkannt. Die Bosch-Gruppe bietet mit der „Bosch IoT Suite“ bereits eine Plattform, die laut eigenen Angaben *„alle Kernanforderungen im IoT abdeckt“* [19]. Ziel ist es, Unternehmen in der Implementierung neuer Services im Netz der Dinge zu unterstützen.

Kunden der Suite könnten vor allem Unternehmen der Autobranche sein. Analog zu Smart Home können sich Smart Cars in der Zukunft sogar selbst fahren, wie Google mit dem Driverless Car zeigt. Ein weiteres Beispiel ist die App Mojio, die dem Fahrer in Echtzeit Daten zum Auto auf das Smartphone liefert. Hiermit sollen sogar potenzielle Schäden vorgebeugt werden. Die Innovation des IoT kann für die Industrie revolutionär sein. Deshalb erwähnen Politiker und Vertreter der Industrie oftmalig den Begriff Industrie 4.0, wenn es um das Netz der Dinge geht. Wenn Firmen heute ihre Geräte vernetzen wollen, sehen sie sich aber noch einem großen Problem gegenüber – der Zusammenführung verschiedener Standards. Dem will das Start-Up-Unternehmen CloudRail entgegenwirken. Durch ihre API können verschiedene Systemstandards vereinigt werden.

Im ersten Kapitel dieser Arbeit wird zunächst das IoT nach dem aktuellen Entwicklungsstand evaluiert. Hierbei werden technische Hintergründe näher gebracht und bereits realisierte Projekte vorgestellt. Weiterhin werden die Industrie 4.0 und Smart Home als Anwendungsbereiche des IoT präsentiert. Der technische Aufbau von APIs und Beispiele bilden den Abschluss des Kapitels.

Des Weiteren wird in der vorliegenden Arbeit die betreuende Firma CloudRail vorgestellt. Unter anderem wird in diesem Kapitel das Produkt, die CR-API, aufgeführt und gezeigt, wie sie das Internet der Dinge antreibt.

Zuletzt folgt die Durchführung des Projektes. Für CloudRail fertige ich ein Browser-basiertes Dashboard an, welches die Vernetzung verschiedener Dienste und Geräte visualisiert. Ziel hierbei ist es, interessierten Kunden und Entwicklern ein ansprechendes Werkzeug zu bieten, mit dem sie u.a. auf Facebook posten, aktuelle Wetterdaten aus ihrem Ort abrufen, eine Philips Hue Lampe steuern oder Live-Daten aus dem Büro von CloudRail einsehen können. Das Dashboard soll letztendlich Interesse für die API wecken, mit der die verschiedenen Dienste integriert wurden.

Kapitel 1 Das Netz der Dinge

1.1 Definition

> *„The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."* [20]

Bereits 1991 hatte Mark Weiser eine Vision, wie sich die Welt der Computer im 20. Jahrhundert entwickeln wird. Mit obigem Zitat legt er den Grundgedanken des IoT fest – eine Welt in der sich Technologie mit dem realen Leben untrennbar vereint. Die wohldurchdachtesten Technologien seien dabei jene, die ihre Funktion dem Auge entzogen erfüllen. Weiser prägte damit auch den Begriff *Ubiquitous Computing*, der für die allgegenwärtige Existenz von Computern um uns herum steht. Der Benutzer befindet sich im Mittelpunkt, während die Technik für ihn unerkennbar im Hintergrund verschwindet. Dies steht im Gegensatz zum heutigen Verwenden von Bildschirmen und dem damit verbundenen direkten Interagieren mit technischen Geräten. Weiser schreibt weiter:

„*Today's multimedia machines makes the computer screen into a demanding focus of attention rather than allowing it to fade into the background",* [20] und meint damit, dass sich die Technik durch Bildschirme in den Vordergrund drängt.

Während also die Technik heute im Mittelpunkt steht, soll das *Future Web*[1] benutzerzentrierter werden. Die Interaktion mit der Technik verläuft versteckt. Wenn sich Geräte im Smart Home gegenseitig steuern, sind sie zu intelligenten Objekten geworden, die Informationen eigenständig weiterleiten und verarbeiten. Diese intelligenten Objekte sind sogenannte cyber-physische Systeme. Cyber-physische Systeme haben zum einen physische Fähigkeiten, sind zum anderen aber auch in globale Netze eingebettet und können so miteinander kommunizieren. Die Entwicklung des Internets der Dinge wird durch das Interagieren mit und dem Erweitern von der realen Welt wesentlich ermöglicht. [21]

Abbildung 1: Karikatur zum Internet der Dinge [1]

Abbildung 1 zeigt scherzhaft die angedeutete Vereinigung der realen und virtuellen Welt. Während der Besitzer das Haus verlässt, scannt der Kühlschrank seinen Inhalt und gleicht ihn mit seiner Datenbank ab. Als er in seinem Bestand zu wenig Milch feststellt, ruft er dem Besitzer über Lautsprecher hinterher. Der Kühlschrank wird als reales Objekt in ein Netz eingebunden und ist somit Teil der virtuellen Welt geworden (cyber-physisches System). Ganz im Sinn von Weiser verfügt der Kühlschrank dennoch über keinerlei Display und fügt sich in die reale Welt ein.

[1] dt.: Internet der Zukunft

Um zu verstehen, wie das IoT funktioniert, hat das Fraunhofer-Institut die Prinzipien des IoT in vier Grundsätzen zusammengefasst [22]:

- **Speicherung individueller Informationen am Objekt**
  Auf den Geräten werden Daten oder Messungen gesammelt. Am Beispiel von Netatmo[2] wären dies die Gradzahl, CO_2-Werte und Luftfeuchtigkeit.
- **Vernetzung der Objekte**
  Das Netatmo kann mit Endgeräten wie dem Smartphone oder PC, aber auch mit Geräten ohne Display vernetzt werden, um bestimmte Funktionen auszulösen oder Daten darzustellen.
- **Individuelle Entscheidungsfindung auf Basis lokal ausgewerteter Information**
  Sind bspw. die CO_2-Werte zu hoch, kann das Netatmo bestimmte Aktionen bei anderen Geräten auslösen. Denkbar wäre die Öffnung von Fenstern oder Türen, wobei hier an den Sicherheitsaspekt gedacht werden muss. Bspw. öffnen diese nicht, wenn niemand Zuhause ist.
- **Individuelle Services auf Abruf zur echtzeitnahen, ereignisorientierten Steuerung von Prozessen**
  Das Smartphone bietet heute schon eine perfekte Fernbedienung zur Steuerung solcher Geräte. Mit ihm können die Werte des Netatmo durch Anwendungen abgerufen, aber auch gesteuert werden.

Nach den bisherigen Untersuchungen lassen sich für das IoT einige Merkmale festlegen.

Das IoT ist ubiquitär. Demnach ist das IoT permanent um uns herum. Es braucht keinen An- und Ausschalter. Ubiquitous Computing und pervasive computing[3] sind dabei nahezu austauschbare Begriffe. Während das ubiquitäre die erwähnte Allgegenwart beschreibt, wird unter pervasive computing die Einbeziehung aller Geräte verstanden.

Nach Weiser ist die durchdachteste Technologie jene, die nicht mehr sichtbar ist. *Das IoT fungiert versteckt* und wird bestenfalls überhaupt nicht wahrgenommen. Die Hauptfunktion der vernetzten Dinge (Briefkasten als Postablage, Kühlschrank als Kühlbehältnis) wird nicht eingeschränkt oder verändert. Das IoT dient als unsichtbare Erweiterung.

Die vernetzten Dinge benötigen keinen Bildschirm oder sonstige Interaktionsoberflächen. Durch den Einsatz von Sensoren werden gewünschte Informationen automatisch und vom Menschen unabhängig aus der realen Welt gesammelt. *Das IoT handelt also autonom.*

Das IoT arbeitet nutzerzentriert. Der Mensch steht im Mittelpunkt des Systems. Die Technik agiert im Hintergrund und dient ihm als Hilfsmittel.

Physische Objekte, die in die virtuelle Welt eingebunden sind, besitzt nahezu jeder. Ganze Geräte und Dinge, werden ins Netz eingelagert. Als Beispiel dienen Smartphones (internetfähige Handys), Smart TVs (internetfähige Fernseher) und seit Neustem auch Smart Watches (internetfähige Uhren).

[2] Netatmo ist ein Raumklimamesser, dessen Werte online eingesehen werden können
[3] dt. durchdringende Computernutzung

Diesem Trend sollen im IoT aber Dinge folgen, die kein Display haben. Kühlschränke, Feuermelder, Wecker und Thermostate, Kaffeemaschinen, aber auch nicht elektronische Geräte wie Türen, Fenster, Treppen könnten im IoT Anwendung finden. Sei es die Auswertung, wie oft und wann diese geöffnet oder betreten wurden oder eine sicherheitsrelevante Vorkehrung, die Hausbesitzer einsetzen. Jeder gebräuchliche Alltagsgegenstand kann durch Sensoren an ein Netz angebunden werden. Der Begriff Internet of Everything (IoE) ist eine Erweiterung zum IoT. Damit soll ausgedrückt werden, dass jedes Objekt um uns herum smart werden kann. Eine bekannte und frühe IoE-Anwendung war die twitternde Blume pothos[4]. Per Feuchtigkeitssensor gab sie ihrem Besitzer in einem Tweet Bescheid, sobald er sie gießen soll. Sie gab außerdem regelmäßig Meldungen über die aktuelle Bodenfeuchtigkeit und bedankte sich für das Wässern.

Was sich neben Pflanzen heute schon steuern lässt, zeigt das folgende Kapitel über einen Anwendungsbereich des IoT – das Smart Home.

1.2 Anwendungsbereich Smart Home

Während die Eltern im Urlaub sind, schmeißt die Tochter eine ausufernde Hausparty mit Freunden. Als sie ihren kleinen Bruder ins Zimmer sperrt, petzt er die Aktion beim Familienoberhaupt. Dieser nimmt kurzerhand das Smartphone, schaltet das heimische Licht aus und lässt die Raumtemperatur auf einstellige Gradzahlen sinken. Die Party hat damit ein Ende gefunden.

Mit diesem TV-Spot warb die Telekom im Jahr 2015 für die Steuerung des Eigenheims durch eine App. Durch diese konnte der Vater das komplette Zuhause kontrollieren und war selbst einige Kilometer davon entfernt. Mithilfe des Spots wollte die Telekom mehr Aufmerksamkeit für eine Technik erhalten, mit der weit mehr möglich ist, als nur Fremde aus dem Haus zu jagen.

Unter Smart Home versteht man nämlich ein automatisiertes und intelligentes Zuhause. Das Angebot bietet neben der Kontrolle und Steuerung des Eigenheims eine Umgebung, in der sich Küchengeräte automatisch ein- oder ausschalten, mit anderen Geräten kommunizieren und dem Besitzer die Kontrolle des Eigenheims erleichtern soll. Ein Beispiel: Miele stellte auf der IFA[5] 2015 eine Waschmaschine vor, die bei Bedarf Nachfüllprodukte online besorgt. Dazu verbindet sie sich mit dem WLAN und lässt sich mit dem Smartphone steuern. Geht das Pulver zu Neige, sendet die Maschine eine Nachricht an den Besitzer. Über eine App lassen sich dann die Produkte schnell und einfach einkaufen.

Die Telekom bietet bereits ein ganzes Heimnetz, in dem sich Thermostate, Kameras, Bewegungsmelder, Rauchmelder und vieles mehr über eine App oder Fernbedienung steuern lassen. Angekommen ist dieses Konzept aber noch nicht, obwohl die Telekom viel Geld in das Marketing steckt. Laut einer Umfrage der Firma lehnen es 42% der Befragten ab, registriert zu werden, wenn sie nach Hause kommen. Weitere 38% wollen die Kontrolle über ihr Heim behalten und Heizungen selbst einstellen. Etwa ein Viertel der Befragten hat Sorge, dass die Daten in falsche Hände geraten. [23] Die fehlende Akzeptanz ergibt sich aus der Angst vor Datenmissbrauch und der Übernahme des Eigenheims durch Fremde.

[4] https://twitter.com/pothos

[5] Die internationale Funkausstellung (IFA) findet einmal im Jahr in Berlin statt

Die Gewährleistung von Datenschutz und Privatsphäre sind, wie in allen vernetzten Umgebungen, eine Hürde, die das Smart Home zu meistern hat.

Als zweitgrößter Energieversorger im deutschen Raum bietet auch die Rheinisch-Westfälische Elektrizitätswerk AG (RWE) ein intelligentes Zuhause an. Über eine Benutzeroberfläche lassen sich ausgewählte Gegenstände der Wohnung fernsteuern.

Abbildung 2: Benutzeroberfläche der RWE Smart-Home-Anwendung [2]

Aufgeteilt nach einzelnen Räumen können Daten von Geräten eingesehen und kontrolliert werden (siehe Abb. 2). Laut RWE lassen sich die Geräte ohne technische Vorkenntnisse schnell installieren und in das Netz einbinden.

1.2.1 Geräte des Smart-Home-Verbundes

Wie die Firmen Telekom und RWE zeigen, können die verschiedensten Geräte zur Smart-Home-Anwendung genutzt werden. Im Folgenden sind Geräte aufgelistet, die für die Implementierung in eine Smart-Home-Umgebung bereits genutzt werden [24]:

- Heizkörperthermostat: Lässt die Raumtemperatur aus der Ferne regeln
- Energy Controller: Wertet die Daten aus einem Stromzähler aus und liefert dem Nutzer eine Verbrauchs-Übersicht
- Raumthermostat: Bildet die zentrale Steuerung von Fußbodenheizungen oder Heizkörpern
- Zwischenstecker: Misst den Stromverbrauch angeschlossener Geräte, z.B. der Kaffeemaschine
- Wandsender: Verfügt über die Möglichkeit mit nur einer Taste verschiedene Aktionen zu steuern, z.B. das Ausschalten aller Lampen im Netz
- Bewegungsmelder: Erkennt Bewegungen in der Nähe und kann verschiedene Aktionen auslösen, wie das Anschalten von Lampen
- Rollladensteuerung: Lässt Rollläden aus der Ferne steuern
- Netatmo: Misst das Raumklima (Temperatur, Luftfeuchtigkeit, CO_2-Gehalt)

- Rauchmelder: Kann einen Feueralarm durch verschiedene Aktionen auslösen, z.B. eine Nachricht an den Besitzer
- Tür-/Fenstersensor: Zeigt an, ob Türen offen oder geschlossen sind
- Samsung Smartcam: Überträgt Livebilder des Zuhauses auf die Benutzeroberfläche
- Yale ENTR: Lässt Türen ohne Schlüssel aus der Ferne schließen
- Philips Hue: Ist eine regelbare LED-Lampe
- Google Nest: Lernt die Raumtemperaturen der Bewohner und reguliert diese automatisch

1.2.2 Arten der Kommunikation

Die festgelegten Merkmale des IoT verlangen eine Umgebung, in der sich die reale und virtuelle Welt vereinigen. Wie aber spricht man Geräte ohne interne Logik an? Wie kommunizieren die Geräte untereinander? Um das Bedienen von Geräten durch den Menschen überflüssig zu machen, gibt es mehrere Konzepte. Diese werden folgend vorgestellt.

WLAN besitzt bereits einen großen Bekanntheitsgrad. In dieser kabellosen Erweiterung zum klassischen LAN (Local Area Network) können Geräte miteinander kommunizieren, ohne an eine Dose angeschlossen sein zu müssen. WLAN bietet somit den ersten Schritt für eine unsichtbare Kommunikation zwischen Geräten.

Bluetooth Low Energy, auch bekannt als Bluetooth Smart, gewinnt heute durch die zunehmende Popularität von Wearables[6], Sportsensoren und Smart Watches an Bedeutung. [25] Das Konzept von Bluetooth sieht vor, Geräte kabellos zu verbinden. Wie beim WLAN, kommunizieren die Geräte über Funk. Bluetooth ist immer dann die erste Wahl, wenn sich zwei Geräte verbinden wollen, die sich in unmittelbarer Nähe befinden. WLAN ist zu bevorzugen, wenn eine höhere Reichweite und höheres Tempo benötigt wird. [26] Bluetooth Low Energy soll dabei das Problem des hohen Energieverbrauchs von Bluetooth lösen.

Ein sehr beliebtes Funkprotokoll im Bereich der vernetzten Geräte ist ZigBee. Entstanden aus einem Verbund von Firmen, die sich auf einen Standard einigen wollten, findet ZigBee beispielsweise Anwendung bei der Kommunikation zwischen der steuerbaren Hue-Lampe und Bridge. Auch das Nest Thermostat baut auf ZigBee. Probleme entstehen nun durch die große Anzahl an Standards, die ZigBee vereint. Es gibt nicht die eine Lösung, sondern mehrere Teillösungen. Mit ZigBee 3.0 sollen jedenfalls 2016 die Standards unter einen Hut gebracht werden, wodurch ZigBee auch zukünftig im Funkbereich etabliert sein wird. [27] Im Januar 2016 gab ZigBee eine Pressemitteilung heraus, in der eine Partnerschaft mit der Thread-Gruppe angekündigt wurde. Diese zeichnet sich durch eine Kompatibilität im drahtlosen Funkbereich mit dem Internetprotokoll (IP) der Thread-Gruppe aus.[7]

[6] Wearables sind elektronische Geräte, die am Körper befestigt werden können. Mehr unter Kapitel 1.6

[7] Die offizielle Pressemitteilung auf Deutsch kann unter http://www.zigbee.org/?wpdmdl=4200 heruntergeladen werden.

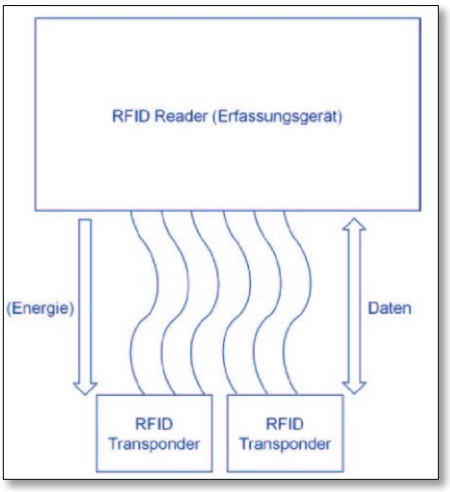


Abbildung 3: Das RFID- System [3]

RFID identifiziert Objekte über Radiowellen. Zwei Komponenten sind hierfür notwendig: ein Radiowellen-Transponder und ein Reader, das Erfassungsgerät. Der Transponder ist dabei in der Lage, sowohl Daten zu senden (Transmitter), als auch zu empfangen (Responder). Er besteht aus einem Chip und einer Antenne. [3]

Die nebenstehende Abbildung 3 zeigt den Aufbau des Systems. Das Erfassungsgerät sendet elektromagnetische Wellen aus. Die Transponder benötigen dadurch keine eigene Spannungsversorgung, sondern nutzen die Energie aus den Wellen[8]. Durch die Wellen werden außerdem Daten der Transponder ausgetauscht.

Wie in allen Funknetzwerken hat auch die RFID den Nachteil der Reichweite. Je nach Umgebung und Frequenzbereich reicht dieser von wenigen Zentimetern bis hin zu einigen Metern.

Anwendung findet das System vor allem in Bereichen, in denen Objekte nicht selbstständig zur Kommunikation fähig sind. Schon lange wird RFID deshalb in der Logistik zur Paketregistrierung benutzt oder bei Eintrittskarten als Zugangskontrolle und Zeitmessung.

Im Smart Home können die vorgestellten Kommunikationsformen zu folgenden Einsätzen kommen:

- **Lokalisierung und Identifizierung von Objekten im Smart Home Netz (WLAN, Bluetooth)**
  Das Netz kennt dadurch nicht nur den Ort, sondern auch die Art des Gerätes und kann einen Hausplan generieren (siehe Abb. 2 RWE Smart Home).
- **Ordnungsmanagement (RFID)**
  Der DVD-Berg wächst stetig, das Bücherregal platzt aus allen Nähten und im Speicher sah es auch schon mal besser aus. Wenn alle Objekte im Bestand mit einem Chip ausgestattet werden, hat man einen permanenten Überblick über Bücher- oder Filmtitel, die man nach Genre ordnen könnte und bringt Ordnung in den Speicher, indem z.B. Gegenstände, die bereits ein Jahr unberührt herumlagen, auf eine Entsorgungs-Liste verschoben werden. Darüber hinaus würde die lange Suche nach Gegenständen der Vergangenheit angehören.
- **Automatisierung (ZigBee)**
  Ein Gegenstand bemerkt die Annäherung einer Person/eines anderen Gegenstandes und führt eine Aktion durch. Denkbar wäre, dass die Haustür bei Annäherung des Eigentümers öffnet oder die Garage, wenn das Auto in den Hof einfährt.

[8] Sogenannte passive Energieversorgung; aktive Energie bedeutet eine eigene Stromversorgung

- **Sensorenauswertung (ZigBee)**
  Jegliche Sensoren zur Messung von Daten, wie Temperatur, Luftfeuchtigkeit, aber auch Licht und Bewegung können zu mehreren Aktionen und Analysen führen.

Firmen werden auch in der Zukunft vermehrt auf eine digitalisierte Welt bauen. Während die Privatpersonen noch skeptisch sind, beginnt in der Industrie eine schleichende Revolution – die Industrie 4.0.

1.3 Anwendungsbereich Industrie 4.0

Der Begriff Industrie 4.0 meint die vierte Stufe der industriellen Revolution. Die erste Stufe der industriellen Revolution bezeichnet die Einführung von Wasser- und Dampfkraft in der Produktion. Die zweite Stufe umfasst die Massen- und Bandproduktion mithilfe elektrischer Energie und die dritte den Einsatz von IT und Elektronik zur Automatisierung der Produktion. [4] Hierzu zählen die Entwicklung von Robotik und Mikroprozessoren. Durch die Bezeichnung Industrie 4.0 wird auch klar, welche Tragweite das IoT besitzt. Die voranschreitende Digitalisierung bewirkt, dass sich die Menge der mit dem Internet verbundenen Geräte exponentiell erhöht. Die Zahl der momentan 15 Millionen vernetzten Geräte soll laut Intel bis 2020 auf mehr als 50 Millionen ansteigen. [4]

Unter dem Begriff Industrie 4.0 vereinen sich neue Geschäftsmodelle, Produktionsstrategien und Produkte. Folgend werden die Teilbereiche vorgestellt.

1.3.1 Geschäftsmodelle

Das Zielmodell in Abbildung 4 zeigt skizzenhaft den Aufbau eines Unternehmens in der Industrie 4.0. Angefangen beim Kunden, der ein vernetztes Gerät besitzt, werden Daten durch Sensoren an eine Cloud[9] übertragen. Kurz gesagt können auf dieser Cloud Daten verarbeitet und angezeigt werden. Das Unternehmen stellt auf der Cloud bestimmte Services zur Verfügung. Gleichzeitig wertet das Unternehmen die Daten aus, versucht Vorhersagen zu treffen und dem Kunden Lösungen für mögliche Probleme anzubieten.

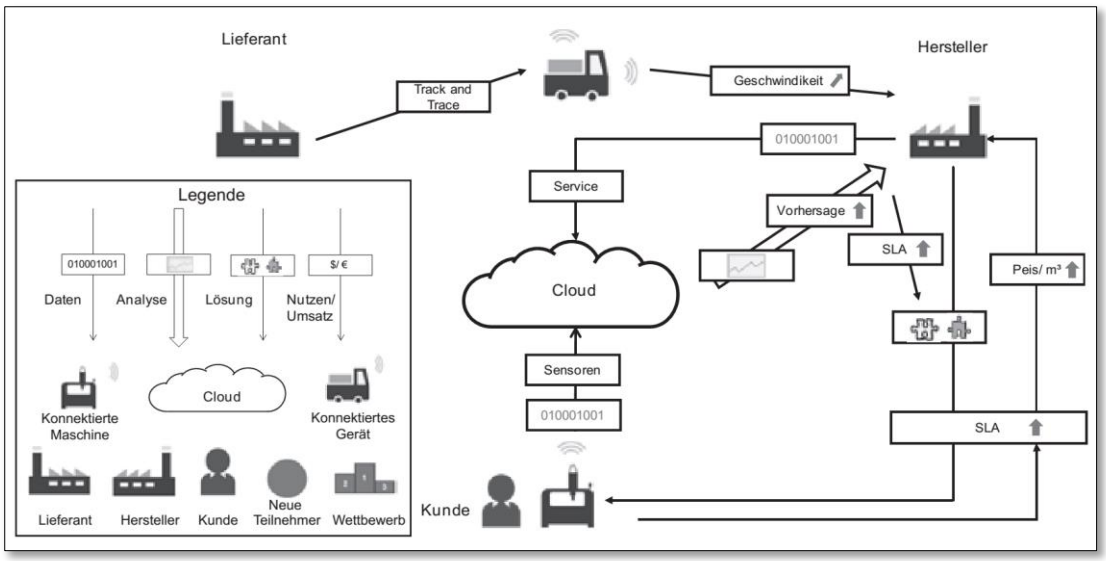


Abbildung 4: Beispiel für ein Industrie 4.0-Zielmodell [4]

[9] Datenwolke, mehr dazu unter Cloud Computing in Kapitel 1.5

Die Kundenbeziehung basiert auf einem Service Level Agreement (SLA)[10]. Mit einem SLA ist gemeint, dass zwischen dem Kunden und Unternehmen eine Vereinbarung über immer wiederkehrende Dienstleistungen besteht. Grundlage sind beidseitige Verpflichtungen, wie die Instandhaltung des Gerätes durch das Unternehmen oder die Bereitstellung der Daten durch den Kunden.
Auf der Lieferantenseite kann eine exakt terminierte Lieferung durch Sendungsverfolgung gewährleistet werden, sodass das Unternehmen längerfristig planen kann.

1.3.2 Produktionsstrategien

Schaut man heute in Firmen, vor allem in die Produktion und Reparatur, stellt man fest, dass Maschinen bedienbare Werkzeuge sind, die von Arbeitern gegebene Aufgaben ausführen. Hier stellt sich der erste Unterschied zur Zukunft ein. Die Mensch-Maschinen-Kommunikation wird eingeschränkt. Die Maschinen nehmen Befehle anderer Maschinen entgegen, die den Menschen komplett ersetzen können. Dieses Prinzip nennt sich M2M (Machine to Machine). Maschinen brauchen keinen Bediener mehr, sondern steuern sich über Sensoren und Software selbst, bzw. gegenseitig.

Ein weiterer Unterschied liegt im Datenbereich. Daten in der Industrie werden in der Zukunft durch die Steigerung der Teilnehmer am IoT nicht nur riesig (Big Data), sondern gewinnen auch an Bedeutung (Smart Data). Damit ist gemeint, dass Maschinen auf der Basis gegebener Daten Entscheidungen finden und treffen, die den Produktionsablauf optimieren. Anhand von Algorithmen erkennen sie beispielsweise Fehlermuster und können daraus Vorhersagemodelle ableiten. [4]
Ein letztes Merkmal der veränderten Produktionsstrategien ist die Augmented Reality[11], die die Erweiterung der realen Welt durch virtuelle Daten beschreibt. In der Industrie 4.0 ist es denkbar, dass Arbeiter, vor allem aus dem Service-Bereich, Aufgaben durch Datenbrillen erleichtert bekommen. Datenbrillen, wie die Google Glass, zeigen Informationen und Anleitungen zu individuellen Bedürfnissen an.
Diese Aspekte sind interessant für die Industrie, denn optimierte Produktionsabläufe (M2M, Smart Data) und bestens ausgestattete Mitarbeiter (Datenbrillen) senken Kosten. Dennoch bleibt die Frage, was passiert, wenn der Mensch in diesem System überflüssig wird.

1.3.2.1 Mögliche Probleme in der Industrie 4.0

> *„Die vierte industrielle Revolution wird die Gesellschaft mindestens so stark verändern, wie es die industrielle Revolution des 19. Jahrhunderts gemacht hat." [28]*

Richard David Precht, ein deutscher Philosoph, sieht in dem neuen industriellen Zeitalter, in der Maschinen Menschen ersetzen werden, einen Umbruch in der Gesellschaft. Die Folgen sind Massenarbeitslosigkeit und Einbruch der Binnenmärkte. Doch wieso ist gerade die Industrie, die von einem funktionierenden Markt profitiert, interessiert an einer solchen Wandlung?

Die im vorherigen Kapitel aufgeführten Aspekte der Optimierung der Produktion bergen viele Chancen. Geringere Kosten, schnellere Ausführung und durchgehende Arbeitsphasen sind nur wenige davon.

---

[10] dt.: Dienstleistungsvereinbarung
[11] dt.: erweiterte Realität

Intelligente Maschinen verändern unsere Industrie. Sie haben Einfluss darauf, wie effizient, wirtschaftlich und ressourcenschonend wir produzieren.

Unternehmen müssen in der Zukunft abwägen, ob sich die Vorteile einer Produktion, in der nur Maschinen arbeiten, für sie auszahlen. Sollten sich Unternehmen gegen den Mensch als Arbeitskraft entscheiden, wird sich ein Umbruch in der Gesellschaft ergeben. Es werden Probleme in der Politik und Wirtschaft entstehen, deren Ausmaße heute nur schwer vorauszusagen sind. Telekom-Chef Timotheus Höttges erwartet mit der Digitalisierung einen Wegfall an Arbeitsplätzen und fordert für die Zukunft ein *„bedingungsloses Grundeinkommen […], um ein menschenwürdiges Leben zu führen"* [29].

Sicher ist, dass mit der Industrie 4.0 ein industrielles Zeitalter anbricht, in dem der Mensch seinen Stand als ausführende Arbeitskraft verlieren kann. Der Service-Bereich (Überwachung, Wartung, Optimierung von Maschinen) wird hingegen an Bedeutung gewinnen.

1.3.3 Produkte

Die Produkte in der Industrie 4.0 zeichnen sich durch die Vernetzung mit anderen Geräten im IoT aus. Als Beispiel wird in diesem Abschnitt die App Mojio vorgestellt.

Mojio (Abb. 5) stellt Autobesitzern einige interessante Funktionen zur Verfügung. Mit der App lassen sich nicht nur Wartungsinformationen, Fahranalysen und Lokation abrufen, sondern auch Live-Diagnosen des Autos durchführen. Um Mojio zu nutzen, muss man laut Hersteller lediglich ein Gerät an den im Auto üblichen OBD 2-Stecker anschließen. Das Gerät lässt sich über eine 3G[12]-Verbindung vom Smartphone, der Smartwatch oder anderen Endgeräten ansprechen. [30]

Abbildung 5: Mojio-Stecker und Applikation [5]

Nicht nur der Diebstahlschutz, sondern auch der Wartungsaspekt sind Vorteile, die Mojio mit sich bringt. Autobauer auf der Welt werden aber nicht lange auf sich warten lassen. Denkbar ist, dass ein Gerät wie Mojio schon bald überflüssig wird und Hersteller eine solche Technologie direkt in den Bordcomputer einbauen.

Und weiteres ist denkbar. Während sich vernetzte Autos heute durch Musikstreaming, Verkehrsmeldungen und GPS-basierte Kartierung auszeichnen, sollen sich zukünftige Smart Cars selbst analysieren und reparieren oder wie Google es vorhat, selbst fahren. Der Mensch wäre lediglich Passagier und müsste dem Bordcomputer das Ziel vorgeben.

Damit ein Auto autonom fährt, sind folgende Faktoren nach Miller essenziell [31]:

- Eine 360°-Kamera, um alle Seiten des Autos im Blick zu haben
- ein flexibler Tempomat, um die Geschwindigkeit im Stau zu regulieren
- Notbremse und Lenkunterstützung, um Kollisionen zu vermeiden
- GPS, um das Auto präzise zu orten und Routen festzulegen
- Laser illuminated Detection and Ranging (LiDAR), um den Abstand zwischen dem Auto und anderen Objekten zu bestimmen

Auch Google setzt in ihrem Driverless Car auf das LiDAR-System. LiDAR ist ein sehr exaktes Entfernungsmesssystem. Durch Laser werden Objekte angestrahlt. Die reflektierten Strahlen werden gemessen und in ein Gesamtbild eingefügt. So kann das Driverless Car ein 3D-Modell seiner Umwelt generieren. Anders als bei Fahrerassistenzsystemen, ist bei den Driverless Cars kein Eingriff von Menschen vorgesehen. Google setzt dabei nicht auf die von Miller aufgeführten Punkte des Tempomats und der Notbremse. Laut Google ist ein Auto erst dann völlig selbstautonom, wenn der Fahrer zu keinem Zeitpunkt die Kontrolle übernehmen muss. [32]

[12] 3G beschreibt die dritte Generation des Mobilfunkstandards

1.4 Plattformen im DIY-Sektor

Anwendungen im IoT sind heute dank vorinstallierten Apps und bereitgestellten SDKs für Nutzer leicht zu implementieren. Während die WunderBar und botanicalls SDKs mitliefern, sind der Arduino und Raspberry Pi vollprogrammierbare Mikrocontroller. Mit den Plattformen wird ein wichtiger Bereich im IoT eröffnet – Do it yourself (DIY), mach es selbst. Nie war es einfacher eigene Anwendungen mit realen Sensoren und Aktoren zu verbinden. Die WunderBar wird mit ihren Sensoren Bestandteil meines Dashboards und deshalb in Kapitel 4 ausführlicher erläutert. Die restlichen drei Plattformen werden folgend im Vergleich vorgestellt.

1.4.1 botanicalls

Abbildung 6: Die botanicalls-Platine [6]

Im Einführungskapitel zum Internet der Dinge wurde bereits die Pflanze pothos erwähnt, die per Twitter über ihren Wasserstand Auskunft gibt. Botanicalls (Abb. 6) spezialisiert sich auf die Kommunikation zwischen Mensch und Pflanze und bietet einen Mikrocontroller, der lediglich auf das Auswerten des Feuchtigkeitssensors programmiert ist. Nachdem die Platine zusammengebaut und das Ethernet-Kabel verbunden wurde (die Platine bietet kein WiFi-Modul), kann sie in Betrieb genommen werden. Durch einen vorkonfigurierten Twitter-Account lassen sich die ersten Ergebnisse ablesen.

1.4.2 Arduino

Botanicalls ist im Gegensatz zum Arduino ein Spielzeug für Pflanzenliebhaber, denn der Arduino ist ein vollprogrammierbarer Microcontroller, der durch eine große Auswahl an verfügbaren Sensoren und Aktoren eine noch größere Bandbreite an Funktionen bieten kann. Er besteht aus zwei Teilen – der Hardware und der Software. Der Hardwareteil ist das Microcontroller-Board (siehe Abb. 7). Ein- und Ausgangsports stellen die Verbindung zur physischen Welt her. Durch digitale und analoge Eingänge können Daten von Sensoren eingelesen werden. Die Ausgänge steuern beispielweise Leuchtdioden, Relais oder Motoren an. Der Softwareteil ist die IDE, die Entwicklungsumgebung. Diese erstellt die Programme, die vom Hardwareteil ausgeführt werden. [33]

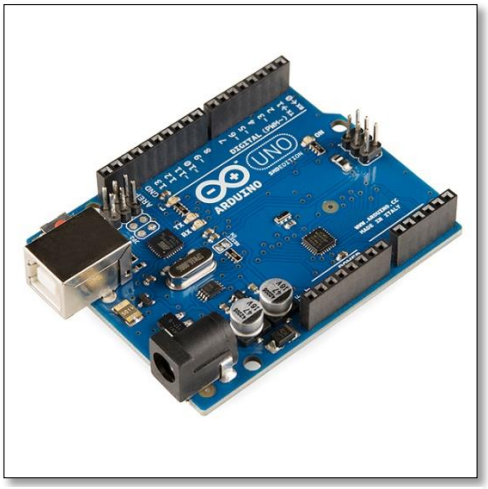


Abbildung 7: Die Arduino Uno-Platine [7]

Programme, sogenannte Sketches[13], werden auf dem Arduino in der Sprache C/C++ verfasst. Ein typisches Programm besteht aus zwei Methoden: Dem `void setup()` für die Initialisierung und einmalige Ausführung der Anwendung und `void loop()` für wiederholende Abläufe. Nachdem der Code auf das Board mittels USB-Anschluss hochgeladen wurde, werden die Funktionen nacheinander ausgeführt.

1.4.3 Raspberry Pi

Das Raspberry Pi (siehe Abb. 8) bietet einen vollprogrammierbaren Mini-Computer. Gängige Sprachen sind Scratch oder Python. Nach letzterem wurde der Beiname Pi abgeleitet. Im Gegensatz zum Arduino ist das Raspberry (in der neusten Generation) durch einen HDMI-Ausgang und auf ARM basierenden Anwendungsprozessor an Bildschirme anschließbar. Der USB-, Ethernet, Video- und Tonanschluss machen das Raspberry zu einem richtigen PC. Da der interne Chip ein ganzes System ausführen kann, basiert das Raspberry auf SoC (System on a Chip). Von der SD-Karte können passende Betriebssysteme wie Linux, aber auch Android starten. [34]

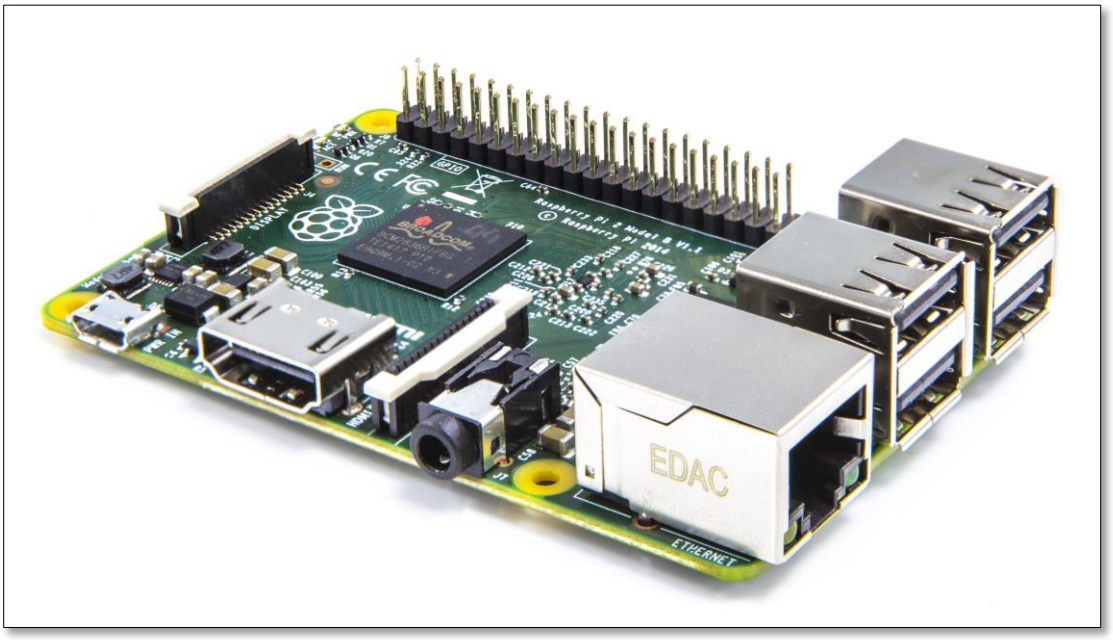


Abbildung 8: Das Raspberry Pi [76]

[13] dt.: Skizzen

1.5 Cloud Computing

Für IoT-Anwendungen über eine der vorgestellten Plattformen gibt es entweder Frameworks des Herstellers (WunderBar, botanicalls) oder externe Angebote (Microsoft Azure), die die Fertigstellung durch bereitgestellte Dienste unterstützen sollen.

Diese Dienste lassen sich im Bereich des Cloud Computing in drei Ebenen spezifizieren. Cloud Computing selbst ist, kurz gesagt, das Bearbeiten von Daten in einem entfernten Rechenzentrum. Es gewinnt an Bedeutung, da durch zunehmende Digitalisierung immer mehr Daten im Netz verarbeitet werden (Big Data). Das bekannteste Beispiel hierfür ist Dropbox, eine Speicherplattform[14] im Netz, auf die von überall zugegriffen werden kann. Das Cloud Computing beschreibt aber nicht nur das Speichern von Daten, sondern bietet in manchen Fällen sogar mehr, wie bspw. eine Entwicklungsumgebung, wie das 2010 veröffentlichte Microsoft Azure. Microsoft stellt hierbei netzbasierte Frameworks zur Verfügung, die Nutzer in der Entwicklung von Web-Apps, virtuellen Maschinen, Datenbanken und mobilen Back-Ends[15] unterstützen.

> *„Mit dem Cloud Computing steht eine ständig veränderbare und geografisch unabhängige Plattform für das Angebot an Rechenleistung zur Verfügung, die für unterschiedliche Anwendungen genutzt werden kann."* [8]

Individuelle Skalierbarkeit, Flexibilität und optimierte Investitionskosten sind nur wenige Beispiele, die das Cloud Computing als Vorteile mit sich bringt.

Die Reichweite an Dienstleistungen misst dabei von dem Angebot einer Infrastruktur (Infrastructure-as-a-Service), einer Plattform (Platform-as-a-Service) bis hin zu einer vollwertigen Applikation (Software-as-a-Service) Für jeden Bereich sind einige Anbieter vorhanden (siehe Abb. 9).

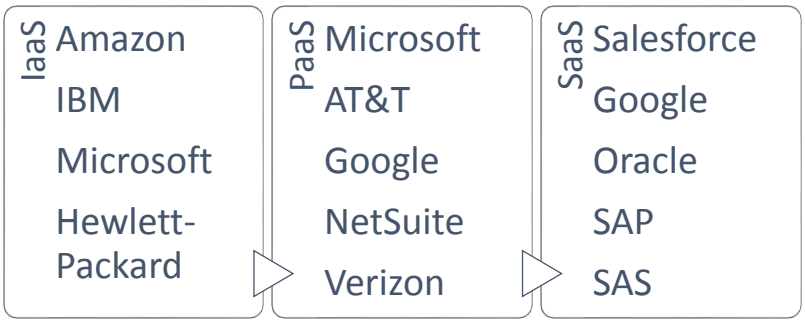


Abbildung 9: Übersicht der wichtigsten Anbieter für Cloud Computing; nach [8]

IaaS-Anbieter stellen Hardwareressourcen als Dienstleistung über Clouds zur Verfügung. [8] Dabei fungiert die IaaS als virtuelle Maschine für mehrere Betriebssysteme.

[14] sog. Cloud Storages
[15] dt. „hinteres Ende" beschreibt die Verarbeitungsschicht in Programmen, dagegen steht das Front-End („vorderes Ende") für die Eingabeschicht des Nutzers

Die Amazon Web Services (AWS) sind hier als Beispiel zu nennen. Sie bestehen aus vier Komponenten:

- dem Simple Storage Service (S3) – **Speicherplatz**
- der Elastic Computer Cloud (EC2) - **Rechenleistung**
- der Simple DB (SDB) – **Datenbank**
- dem Simple Queue Service (SQS) – **Nachrichtendienst für die Services**

Vorteil der Nutzung ist, dass lediglich die in Anspruch genommenen Funktionen in Rechnung gestellt werden. Der bekannteste Kunde der Web Services ist Netflix. „*AWS ermöglicht Netflix, innerhalb kürzester Zeit Tausende Server und mehrere Terabyte Speicher bereitzustellen.*“ [35] So können die Nutzer überall auf der Welt Film- und Serieninhalte streamen.

Die PaaS stellt Laufzeit-/Entwicklungsumgebungen, wie das angesprochene Microsoft Azure, zur Verfügung. Sie dient also der Entwicklung und Implementierung von Anwendungen. Ein anderes Beispiel für PaaS ist ThingSpeak. ThingSpeak ist durch eine kostenlose Anmeldung online zugänglich. [36] Es lässt gesammelte Daten in einer Cloud speichern, um dort von Programmen ausgewertet und in Aktionen umgewandelt zu werden. Es werden Arduino, Raspberry Pi, sowie mehrere Web-Anwendungen wie Twitter unterstützt. Wer einen Arduino besitzt, benötigt also nicht zustätzlich eine botanicalls-Platine, um Pflanzen sprechen zu lassen. Durch einen Feuchtigkeitsmesser kann der Wasseranteil gemessen, auf ThingSpeak analysiert und in einen Tweet verpackt werden.

Bei SaaS handelt es sich um bereitgestellte Softwares, die auf Servern ausgeführt werden. Marktführer im Bereich der Unternehmenssoftwares ist SAP. Mit SAP können Unternehmen spezifische Daten beispielsweise über interaktive Dashboards abfragen.

Abbildung 10: Das SAP Business One Dashboard [9]

Abbildung 10 zeigt ein solches Dashboard. Durch dieses können Unternehmen Daten analysieren, strukturieren und in Diagrammen darstellen lassen. Produktivitäts-Tools geben durch Reporte Auskunft über Echtzeit-Daten. [9]

Firmen versprechen sich so schnellere und bessere Entscheidungsfindungen, erleichterten Zugang zu wichtigen Daten und effizienteres Arbeiten.

1.6 Wo die Technik heute steht

Das Internet der Dinge kann in jedem Lebensbereich Einsatz finden. Smart Home und die Industrie 4.0 sind zwei davon. Prominentes Beispiel für Realisierungskonzepte sind die in Kapitel 1.2.2 aufgeführten RFID-Chips. Heute existieren bereits solche Chips mit interner Logik. Gerade in Warenwirtschaftssystemen können RFID-Chips neue Automatisierungstechniken ermöglichen. Volkswagen beispielsweise verwendet die Chips, um falschen Zuliefererteilen auf die Spur zu kommen. Der Paketdienst FedEx kann permanent jedes Paket orten. Movie Gallery, ein Film-Verleih, spart Arbeitsstunden, indem monatliche Bestandsaufnahmen reduziert werden können. Der Begriff *now-economy* beschreibt, dass Ort und Zustand von Gütern den Unternehmen in Echtzeit zur Verfügung stehen. [37]

Ein weiteres Beispiel für vernetzte Dinge findet sich im Konsumbereich. Sogenannte Wearables sind am Körper befestigte, tragbare Sensoren oder Geräte. Zwei Wearables wurden bereits kurz erwähnt – die Smartwatch und die Datenbrille. Ein bereits etabliertes Wearable ist das Fitness-Armband. Dieses soll Menschen helfen, gesünder und sportlich aktiver zu leben.

Abbildung 11: Polar Loops [10]

Fitnessarmbänder wie der Activity Tracker von Polar (Abb. 11) zeigen die gemachten Schritte, errechnen die verbrauchten Kalorien und bereiten alles in übersichtlichen Grafiken auf. Darüber hinaus gibt die Anwendung Tipps für das Erreichen von weiteren sportlichen Zielen. Durch die Sensoren kann das Armband sogar feststellen, ob der Nutzer schläft und dabei die Schlafdauer in die Auswertung einbeziehen. Die Synchronisation von Armband und Endgerät erfolgt über Bluetooth. [38]

Laut einer Studie steigt die Bekanntheit von Wearables in Deutschland. Im Sommer 2015 verzeichnete der Anstieg ganze 8%. Tatsächlich nutzen wollen das Ganze momentan aber nur 3% aller Deutschen. Auch hier spielt für viele der Datenschutz eine Rolle. Mehr als ein Viertel der Befragten ist gegen die Überwachung ihrer Daten. [39]

Gewöhnliche Dinge wie das Fitnessarmband verfügen über die Fähigkeit, Auswertungen anzustellen oder durch den RFID-Chip über eine interne Logik. Die Anwendungen zeigen, dass im Internet der Dinge alle herkömmlichen Gegenstände Platz finden können. Der Datenschutz wird sich dagegen nie komplett gewährleisten lassen, auch wenn dies nötig ist, um die Akzeptanz aller Konsumenten zu erhalten.

1.7 Blick in die Zukunft

In der Zukunft wird sich die Digitalisierung weiter auf unser Leben auswirken. Frei nach den Zuboffschen Gesetzen gilt [40]:

- Was digitalisiert werden kann, wird digitalisiert.
- Was automatisiert werden kann, wird automatisiert.
- Was überwacht werden kann, wird überwacht.

Für das IoT bedeutet dies, dass mehr und mehr Geräte im Netz einen Platz finden, dass diese sich selbst steuern und im Netz zurecht finden und dass diese zwar physisch verschwinden, aber im Internet immer noch überwacht werden können. Der NSA-Skandal 2014 und die Nachrichten über gehackte Server von Großfirmen sind nur zwei Beispiele für die nicht zu gewährleistende (Daten-)Sicherheit im Netz.

Wenn sich das IoT bei den heute skeptischen Menschen durchsetzen soll, müssen Gesetze über Datennutzungsrechte herausgebracht werden. Erst wenn ein Vertrauen in das System geschaffen werden kann, wird sich das IoT nicht nur in der Industrie entfalten.

Die Chancen, das IoT zu realisieren, sind heute nämlich so gut wie nie zuvor [41]:

- **Sensoren werden immer billiger**
  Die Preise für Sensoren fielen in den letzten 10 Jahren um durchschnittlich 54%. Grund hierfür sind optimierte Produktionsverfahren und fallende Ressourcenpreise.
- **Günstige Bandbreite**
  Auch die Kosten für Bandbreite verringern sich stetig. Im letzten Jahrzehnt nahmen diese um das 40fache ab. Das Internet wird darüber durch Glasfaserkabel schneller, es fasst also mehr Bits pro Sekunde als vorher.
- **Niedrige Prozessorkosten**
  Bei Prozessoren ist der Einbruch am stärksten – im Schnitt fielen die Kosten um das 60fache in den letzten 10 Jahren.
- **Smartphones**
  Smartphones sind ein optimales Endgerät für das IoT. Sei es im Smart Home Bereich als Fernbedienung oder im Gesundheitsbereich als Display für Fitnessarmbänder und andere Wearables. Zudem gewinnen Smartphones an Bedeutung: Im zweiten Quartal 2015 waren etwa 74% aller verkauften Handys Smartphones. Im vergleichbaren Vorjahresquartal lag diese Quote bei 65%. [39]
- **Allgegenwärtiges Wi-Fi**
  Kosten für Internet im Zuhause oder unterwegs sind heute niedrig und werden weiter sinken. Im asiatischen Raum sind kostenlose Hotspots[16] keine Besonderheit. In Deutschland bieten viele Großstädte bereits flächendeckendes Wi-Fi für umsonst.

[16] Orte mit freiem Internetzugang

- **Big Data**
  Der Aspekt des Big Data wurde bereits erklärt. Kurz gesagt bringen mehr Geräte noch mehr Daten mit ins Netz. Die technischen Voraussetzungen, riesige Datenmengen zu aggregieren, sie zueinander in Beziehung zu setzen und in Echtzeit zu verarbeiten, sind durch Server-Farmen und Softwares vorhanden.
- **IPv6**
  Das IPv6 bietet im Gegensatz zum IPv4, das heute noch gewöhnlich ist, 128-Bit Adressen. Damit löst das IPv6 die Adressknappheit auf der Welt und bietet darüber hinaus eine fast grenzenlose Zahl an Adressen im Netz.

1.8 API – Schnittstelle für Entwickler

In diesem Kapitel wird ein wichtiger Schlüssel im IoT vorgestellt, dessen Entwicklung ebenfalls die Grundsteine für eine vernetzte Zukunft legt.

Diese Entwicklungen sind sogenannte APIs – Programmierschnittstellen, die die Verbindung zwischen Geräten und Webdiensten bilden.

Die Elektronik-Firma Best Buy, die mit ihrer Best Buy Open API auf dem Markt ist, zieht in einem Video einen interessanten Vergleich. [42] Der Unterschied zwischen herkömmlichen Programmen und APIs liegt in der Entwicklung. Die offene Entwicklung (open development) führt zu offenen Datensystemen, den APIs. Die geschlossene und traditionelle Software-Entwicklung führt zu geschlossenen Programmen. Traditionelle Softwares sind demnach zu vergleichen mit Brettspielen. Sie sind nützlich für eine Dauer, aber sobald die Spiele die individuellen Bedürfnisse nicht mehr erfüllen oder nicht mehr altersgerecht sind, lassen sich die Spielanleitungen nicht ändern. Neue Brettspiele müssen angeschafft werden. APIs sind wie ein Deck von Karten. Mit Karten können Kinder und ältere Menschen spielen. Dies liegt an der Fülle von Möglichkeiten, die mit Karten zur Verfügung stehen. Allein Poker hat Millionen Varianten. Karten erweitern Spiele oder lassen sich durch andere Spiele erweitern. Die Spielanleitung, im übertragenen Sinn die Funktion der Software, ist veränderbar, sowie stetig ergänzbar. Das macht APIs aus.

Im allgemeinen Verständnis sind APIs überall zu finden. In der Spieleentwicklung gibt es Grafik-APIs, im Internet Web-APIs und in Firmen Datenbanken-APIs. Tatsächlich ist die Entwicklungsumgebung jedes Programmes auf Windows, das von externen Herstellern erstellt wurde, über die WinAPI bereitgestellt worden. Ein weiteres Beispiel sind Apps wie Netflix, die durch die Bereitstellung von APIs auf allen Betriebssystemen der Laptops, Tablets und Smartphones aufgerufen werden können. [43]

Die API ist also eine weit verbreitete Schnittstelle, die externen Nutzern die Anbindung an ein System ermöglicht.

Um dies genauer zu verstehen, wird der Begriff Framework[17] herangezogen. Ein Framework bietet Programmierern eine wiederverwendbare Struktur und ist selbst keine vollständige Applikation.

[17] dt.: Programmiergerüst

> *„You customize a framework to a particular application by creating application-specific subclasses of abstract classes from the framework“.* [44]

Gamma, Helm, Johnson und Vlissides meinen mit dem Satz, dass ein Framework Abhilfe durch vorhandene Bibliotheken schafft. Eine Charakteristik des Frameworks ist somit die Inversion of Control (dt. Umkehrung der Kontrolle), womit ausgesagt werden soll, dass der dargelegte Kontrollfluss aus wiederverwendbaren Klassen stammt. [45] *„The framework dictates the architecture of your code“.* [44] Frameworks haben Einfluss auf Struktur und Design des Codes.

APIs bieten ebenfalls Implementierungsbruchstücke an, wie die Facebook API zeigt. Damit können Web-Entwickler den Like-Button auf ihre Website einbinden oder die Besucher der Seite ihre Gedanken direkt auf Facebook posten lassen. Im Gegensatz zu den meist statischen Frameworks sind APIs stets den Bedürfnissen entsprechend anpassbar. Der Like-Button, der nur einen Link, nämlich die betreffende Seite, referenziert und über ein vorgegebenes Layout verfügt, kann mittels Frameworks realisiert werden. Für den Post-Dialog muss allerdings eine API benutzt werden, da sich die Nutzerinformationen, Inhalt des Posts und Ort des Posts individuell ändern können. Auch ist der benötigte Authentifizierungsprozess durch Frameworks nicht umsetzbar.

APIs bieten demnach Funktionalitäten an und setzen keine Design-Standards fest. In diesem Sinne sind APIs reine Funktionsbibliotheken.

1.8.1 Der Nutzen von APIs

In Kapitel 1.3 wurde bereits erwähnt, dass sich die Anzahl der vernetzten Geräte exponentiell erhöht. Für Menschen ist es in der heutigen Zeit wichtig, überall und zu jeder Uhrzeit Informationen zur Hand zu haben. In der Industrie hat sich ein großer Wandel vollzogen. War in den 1990ern die firmeneigene Homepage die einzige Verbindung mit dem World Wide Web, so sind zukünftig Smartphones, TVs, Autos und weitere Geräte der Kunden mit dem Unternehmen vernetzt. Auch der Bereich des Smart Home gewinnt, wie dargestellt, an Bedeutung. Die Zahl an Anwendungen in diesen Bereichen vergrößert sich rasend und führt dazu, dass der altgediente Internet-Browser unsichtbar wird. E-Mails, Facebook, Twitter, Nachrichten, Musik, Videos, Skype und Netflix – schaut man auf den Gebrauch von alltäglichen Webinhalten, erkennt man, dass die meisten in Form von Anwendungen betätigt werden.

APIs stellen die Wartung, Sicherheit, Updates und Kontrolle von diesen Anwendungen zur Verfügung. [46] Des Weiteren kann die (Weiter-)Entwicklung einer Anwendung in andere Unternehmen ausgelagert werden, sodass Dienstleistungen entstehen. Dieser Dienstleistungssektor spielt mit der zunehmenden Zahl an vernetzten Geräten eine immer wichtigere Rolle. APIs sind demnach eine Notwendigkeit, um dem Trend der datenorientierten Zukunft standzuhalten und schon heute unverzichtbar, wenn es um die Verarbeitung von Anwendungen geht.

1.8.2 Web-API

Um die große Anzahl an APIs einzugrenzen, fokussiert sich die vorliegende Arbeit im weiteren Verlauf auf Web-APIs.

Web-APIs haben vor allem die Eigenschaft, dass sie sich im weltweit größten Netzwerk befinden – dem Internet. APIs im Web ermöglichen Anschluss an einen Web-Service, um diesen in ein System zu integrieren oder zu erweitern. Netflix, Facebook und Twitter sind die bekanntesten Beispiele für Anwendungen, die Web-APIs nutzen.

Netzwerkeffekte ermöglichen dabei eine schnell wachsende Popularität der Web-APIs.

> *„Wenn man Menschen und Ideen miteinander verbindet, entwickeln sie sich weiter. Mehr Teilnehmer erzeugen mehr Wert, der wiederum mehr Teilnehmer anzieht und so weiter."* [47]

Der Netzwerkeffekt ermöglichte erst Plattformen wie Facebook und Twitter den großen Erfolg. Dies führt dazu, dass so gut wie jede Homepage einen integrierten Like- oder Share-Button aufweist, bzw. die Möglichkeit bietet über die Seite zu twittern. Manche Seiten bieten sogar einen optionalen Facebook-Login, damit sich Nutzer nicht neu registrieren müssen. All dies wird über die Facebook- oder Twitter-API realisiert. Die Bedeutung von Web-APIs hat in den letzten Jahren durch die sozialen Medien stark zugenommen.

1.8.3 Infografik zur Web-API

Abbildung 12 auf der nächsten Seite zeigt eine Infografik zum Thema Web-APIs. Die relativen Daten wurden teilweise selbst errechnet und sind nicht eins zu eins wiederzufinden. Als Grundlagen dienten hier die Absolutzahlen der einzelnen Bereiche und die gerundete Gesamtzahl an APIs im Netz, die auf der Plattform `http://programmableweb.com/api` gelistet sind. Die Gesamtzahl beläuft sich auf etwa 14.000. (Stand Oktober 2015)

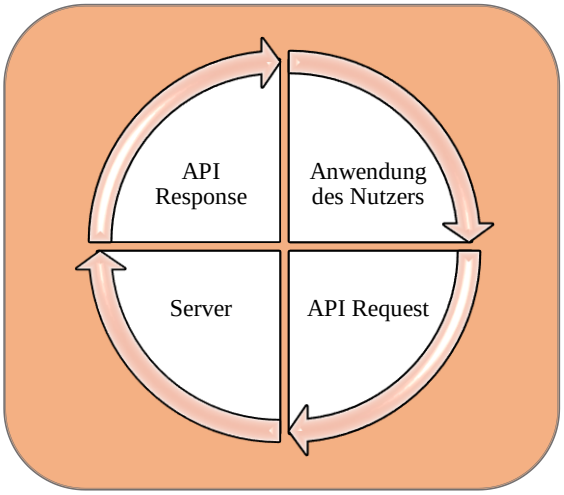
Die API

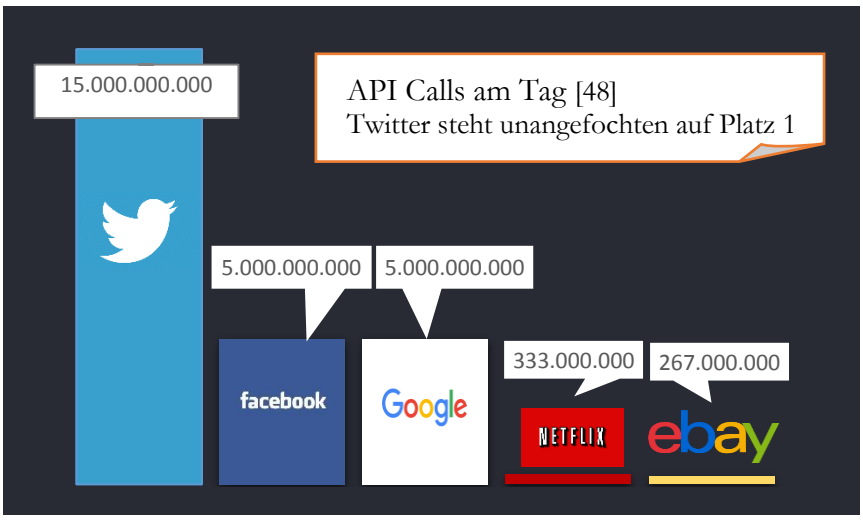
Application Programers Interface

Infografik zum Bereich der Web-API

Definition

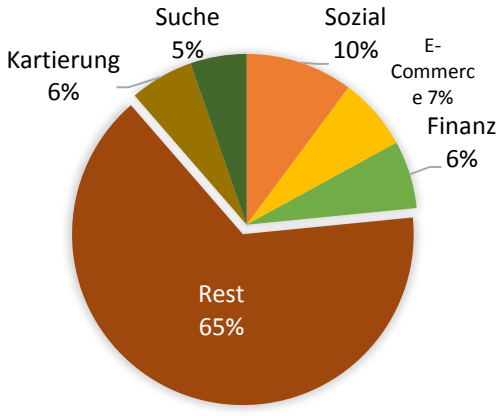
Über Web-APIs können öffentliche Anwendungen im Netz in die eigene integriert werden oder diese erweitern.





Der Prozess eines API-Calls ähnelt dem des HTTP. Die meisten benötigen aber einen Access Token.

Bereits 2000 gab es die erste API von Salesforce, die ermöglichte, Daten auf mehreren Geräten zu teilen. [49]



Die wichtigsten Bereiche [50]

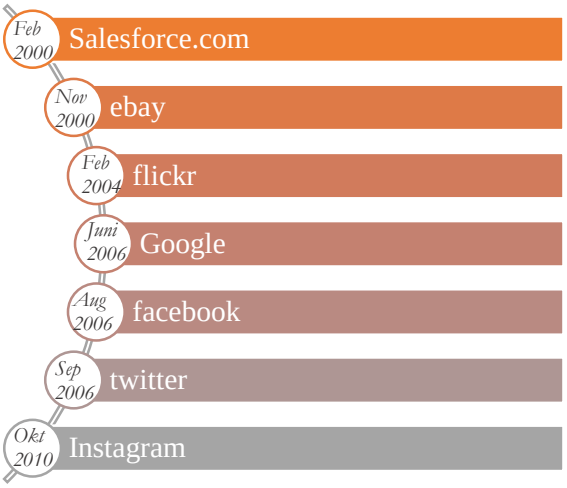


Abbildung 12: Infografik zum Thema Web-API

1.8.4 Der Einsatz der Web-APIs

Im Folgenden werden einige Beispiele für API-Calls der größten Webanwendungen Facebook, Twitter und Google gezeigt.

Facebook
Die Facebook API steht Mitgliedern des sozialen Netzwerks zur Verfügung. Die API ermöglicht

- Anwendungen die Einbindung der sozialen Plug-ins, Fotos, Veranstaltungen, Gruppen, Freunden
- die Veröffentlichung eigener Texte auf der eigenen oder fremden Facebook-Seite
- den optionalen Login auf der externen Seite

Mehr zur Implementierung als Beispiel in Kapitel 1.8.10.

Twitter
Durch die API können Nutzer

- die neuesten Twitter-Nachrichten, Status, Nutzerinformationen abrufen
- die Trends durchstöbern
- den Tweet- und Follow-Button benutzen
- selbst Tweets absetzen

Eine Erklärung für die hohe Zahl an Calls pro Tag (siehe Abb. 12) sind viele populäre externe Anwendungen wie TweetDeck oder TweetBot, die auf die API zurückgreifen.

Google
Google bietet gleich mehrere APIs für jeden Bereich. Mit der Google Drive API können Interaktionen mit dem Cloud Storage durchgeführt werden. Beispiele sind das Hoch- und Herunterladen von Dateien. Durch die Gmail API können Mails versendet werden, durch die Google Play Developer API können Entwickler Android Spiele für den Play Store entwerfen, die Google+ API lässt sozialen Kontext zu und die Translate API bringt den Translator in externe Anwendungen.[18]

[18] Eine große Auswahl an Google-APIs mit Erläuterung gibt es unter `https://developers.google.com/apis-explorer/#p/`

1.8.4.1 Die bedeutendsten APIs nach Kategorien

Folgende Liste zeigt eine kategorisierte Auswahl der bedeutendsten APIs. [50]

- **Kartierung**
  Google Maps: Geocoding[19], Lokalisierung und das Erstellen von Reiserouten mit Distanz und Zeitaufwand sind beispielhafte Features der API
  foursquare: Präsentiert Sehenswürdigkeiten, Restaurants, Einkaufshallen und Tipps für viele Städte

- **Reisen**
  Expedia: Ermöglicht das Buchen von Hotels, Flügen und Autos für den Urlaub

- **Musik**
  Spotify Web: Erlaubt Zugang zu Daten aus dem Spotify Musik Katalog. Hierzu zählen Künstler, Alben und Musikstücke. Außerdem kann der Nutzer seine Playlists und gespeicherte Musik einsehen.

- **Sozial**
  Facebook und Twitter sind hier Spitzenreiter. Ein weiteres Beispiel ist LinkedIn, das soziale Netzwerk für Geschäftsbeziehungen.
  LinkedIn: Bietet die Möglichkeit, die Connections[20], Unternehmen und Jobs anzeigen zu lassen.

- **eCommerce**
  Best Buy Products: Zeigt Produktinformationen, wie Preise, Verfügbarkeit, Bilder und Beschreibungen
  eBay: Zu den Funktionen zählen das Listen von Produkten, das Einsehen von Produktinformationen und Tätigen von (Ver-)Käufen.

- **Sport**
  Fitbit: Ermöglicht die Überwachung persönlicher Daten, wie Gewicht, Aktivitäten und Schlafzeiten. Über die API können die Daten in externen Anwendungen zum Einsatz kommen.

- **Fotografie**
  Instagram: Das Hochladen, Anschauen und Verwalten der Bilder sind Funktionen der Instagram API
  flickr: Funktioniert wie Instagram. Obwohl 6 Jahre früher erschienen, verliert die API durch die Popularität von Instagram an Bedeutung.

- **Finanzen**
  PayPal: Bietet einige Features für Online-Bezahlungen an und ist eine anerkannte Zahlungsart auf vielen Verkaufsseiten.

---

[19] Durch Geocoding können Ortsnamen in Längen- und Breitengrad abgebildet werden. Dementsprechend löst das Reverse Geocoding die Koordinaten in einen Ortsnamen auf.
[20] Connections (dt. Verbindungen) stehen in LinkedIn für Kontakte des Nutzers

Stripe: Das Einsehen des Kontostands, das Tätigen von Überweisungen, das Aufgeben von Daueraufträgen und das Verwalten des Kontos gehören zu den Basisfunktionen der API.

- **Video**
  YouTube: Ermöglicht das Einbetten der Suche nach Videos und den Videos selbst
  Netflix: Bietet Zugang zu allen Netflix-Filmen und Serien.

- **Suche**
  Google Custom Search: Kann benutzerdefinierte Suchoptionen mittels Google durchführen

Um zu verstehen, wie die Web-APIs funktionieren, müssen zunächst Grundlagen des Client-Server-Modells und RESTful HTTP gelegt werden. Am Beispiel der Facebook API werden dann die Web-APIs erklärt.

1.8.5 Client-Server-Modell

Auf dem Client-Server-Modell basieren die meisten Internet-Dienste, wie E-Mail oder Web. Der Client (Auftraggeber) kommuniziert mit Servern (Auftragnehmer) auf der gesamten Welt. Die Regeln und Formate werden von Protokollen übernommen. Im Web ist dies das Hypertext Transfer Protocol, HTTP.
Das Prinzip ist einfach: Der Client stellt eine Anfrage (Request), indem er beispielsweise eine Website aufruft. Der Server antwortet (Response) mit der angeforderten Seite.

1.8.6 Grundlagen von Web-Services

In dieser Arbeit wurde mit den Amazon Web Services bereits ein Beispiel für die Thematik der Web-Services eingeführt. Web-Services integrieren auf dem Web basierende Anwendungen durch mehrere Standards. Sie können durch SOAP, REST, XML-RPC und JavaScript realisiert werden. SOAP und REST werden dabei am häufigsten eingesetzt.
REST ist ein Architektur-Stil für verteilte Systeme und Web-Services, der beschreibt, dass jede Ressource im Netz adressierbar ist. Auf dieser Architektur basiert das Web, denn jede Web-Seite ist durch eine URI, bzw. URL bestimmt. REST nutzt demnach HTTP als Protokoll.

Web-Services, die dem REST Architektur-Stil unterliegen, sogenannte RESTful Web-Services, besitzen folgende Merkmale [51]:

- **Identifikation von Ressourcen durch URIs**
  URIs haben einen globalen Namensraum. Zudem stellt die URI als ID sicher, dass Ressourcen eindeutig adressierbar sind. [52]
- **Standardmethoden**
  Jede Ressource in REST verfügt über den gleichen Satz an HTTP Methoden: GET, POST, PUT und DELETE. Über HTTP-Methoden kann die Art der Anfrage gewählt werden. So verhalten sich beispielsweise der GET- und POST-Befehl bei Serveranfragen unterschiedlich. Während bei GET die übermittelten Parameter in der URL als Query String[21] auftauchen, sendet POST diese versteckt im Body des Requests.
- **mehrere Formate**: XML, JSON, YAML, u.v.m.

SOAP ist ein auf XML-basierendes Protokoll. Als Protokoll definiert SOAP einen Standard für die Integration von Web-Services. Ein direkter Vergleich mit REST ist damit nicht möglich, denn SOAP ist ein konkretes XML-Format und REST ein abstrakter Architekturstil.

REST setzt sich aber mehr und mehr im API-Sektor durch. Die Flexibilität angesichts mehrerer Formate und der bekannte Standard mit HTTP Methoden bieten dem Entwickler Vorteile. Außerdem ist jede Ressource durch die URI eindeutig adressierbar. Da auch die CloudRail-API eine RESTful API ist, werden die Web-APIs folgend bezogen auf die REST-Architektur erklärt.

1.8.7 API Requests

In dieser Arbeit werden beispielhafte API-Requests ausgeführt. Der Aufbau dieser HTTP-Requests wird folgend kurz erläutert:

```
Request-Line:          Methode URL      HTTP/1.1
Request-Header:        Host: Basis-URL
                       Leerzeile
Entity-Body:           Body-Parameter
```

Die Basis-URL ist der Name des Servers, bspw. `http://server.com`, der den Request verarbeitet. Die URL stellt den weiteren Pfad dar, bspw. `/accounts/dirk/profile`. Nach einer Leerzeile folgen die Body-Parameter, also Inhalte des Request.

Als Methode stehen die HTTP-Methoden PUT, POST, GET und DELETE zur Verfügung.

[21] dt.: Abfragezeichenkette

Die Funktion der HTTP-Methoden bleibt bei API-Requests die gleiche:

- **POST fügt eine neue Ressource hinzu**
  Für Web-Anwendungen sind ausnahmslos Profile Voraussetzung für die Nutzung. Diese können durch POST-Befehle angelegt werden. Weitere Beispiele sind das Erstellen von Dateien und Ordnern in Cloud Storages und das Schreiben und Senden von Mails.
- **PUT wird für das Aktualisieren eines bereits vorhandenen Wertes benutzt**
  PUT kann die durch POST angelegten Ressourcen abändern. So sind Profilaktualisierungen einfache Beispiele. Ein weiteres ist die fernsteuerbare Lampe Philips Hue, die per PUT-Befehl die Farbe, Helligkeit und Sättigung ändern kann.
- **GET lässt den Status einer Ressource anfordern**
  GET ist der wohl meistgenutzte Befehl, da jegliche Profilinformationen, Status, herunterladbare Inhalte und sonstigen Daten abgefragt werden können.
- **DELETE löscht vorhandene Ressourcen**
  Der Lösch-Befehl DELETE kann bestimmte Informationen löschen, wie Dateien im Cloud Storage oder einzelne Profilinformationen.

Folgend ein beispielhafter Request, der eine E-Mail-Adresse und einen Nachnamen für das Benutzerkonto des Nutzers Dirk auf dem Server hinzufügt:

```
POST        /accounts/dirk/profile      HTTP/1.1
Host:       http://server.com

email=dirk@mail.com&name=nowitzki
```

1.8.8 API Responses

Die HTTP-Response besitzt folgenden Aufbau:

Status-Line:	*Statuscode*
Response-Header:	*Response-Parameter*
	Leerzeile
Entity-Body:	*Objekt*

Der Statuscode gibt Auskunft, ob der Request erfolgreich (2XX: Success) oder ohne Erfolg (4XX-5XX: Error) war. Response-Parameter können beispielsweise die Location oder Last-Modified beinhalten. Letzteres ist ein Wert der preisgibt, wann die Seite oder Datei zuletzt bearbeitet wurde. Im Entity-Body steht das angefragte Objekt. In den meisten Fällen sind dies Dateien oder Web-Seiten.

Bei APIs werden häufig Werte angefragt oder neu gesetzt. Die Response-Dateien sind entweder im XML- oder JSON-Format. In der Antwort stehen bei PUT, POST und DELETE meist Success oder Error für die Durchführung der Aktion. Bei GET werden die Werte im Format abgerufen.

1.8.8.1 XML

XML (eXtensible Markup Language) zeichnet sich durch sprechende Tags[22] aus.

```
<hochschule>
      <standort>
            <adresse>
            Badstr. 24
            </adresse>
            <plz>
            77652
            </plz>
            <ort>
            Offenburg
            </ort>
      </standort>
</hochschule>
```

Die Zeilen zeigen einen beispielhaften XML-Code. Hochschule ist die oberste Ebene. Darunter fällt der Standort, welcher in Adresse, Postleitzahl und Ort aufgeteilt ist. Der Aufbau ist auf diese Weise hierarchisch strukturiert.

1.8.8.2 JSON

JSON (JavaScript Object Notation) bietet ebenso einen hierarchischen Aufbau. Allerdings erscheint er durch die Reduzierung auf die Hauptbestandteile oft lesbarer als XML. Der Fokus liegt auf der Datenstruktur, nicht auf dem Text.

```
{
      „Hochschule“: {
            „standort“: {
                  „adresse“: „Badstraße 24“;
                  „plz“: „77652“;
                  „ort“: „Offenburg“
            }
      }
}
```

Die Auswahl, ob JSON oder XML, wird von den Entwicklern festgelegt. Im API-Bereich hat sich jedoch die lesbarere Variante JSON etabliert.

1.8.9 Authentifizierung

Das Problem von APIs ist, dass die Authentifizierung der zugreifenden Endgeräte durch eine externe Anwendung geregelt werden muss. Die API Requests müssen allesamt die Login-Informationen für die Anwendung mitschicken, sodass diese den Request für den gemeldeten Nutzer durchführen kann. Dies bedeutet auch, dass die externe Anwendung die eingegebenen Daten auslesen könnte, bzw. ein Hacker leichtes Spiel hat, die Requests zu entschlüsseln. Folgend werden zwei Verfahren vorgestellt, die dieses Problem lösen und im API-Bereich etabliert sind.

[22] dt.: Kennzeichen; bilden die Architektur eines HTML-Dokumentes

1.8.9.1 Das OAuth-Verfahren

OAuth, ein offenes Protokoll zur Absicherung der API-Autorisierung von Webanwendungen, gewährleistet, dass immer der jeweilige Dienst-Anbieter die Zugangsdaten des Benutzers prüft und seine Identität unbekannt bleibt. [53] Ein erster Schritt ist die konsequente Nutzung von HTTPs. HTTPs verschlüsselt die mit HTTP übertragenen Daten, indem SSL/TLS eingesetzt wird. Zunächst werden beide Kommunikationspartner authentifiziert, d.h. sie beweisen, dass sie tatsächlich diejenigen sind, die sie zu sein vorgeben. Anschließend wird ein Schlüssel ausgetauscht, der zur Verschlüsselung der Daten genutzt wird. [54]

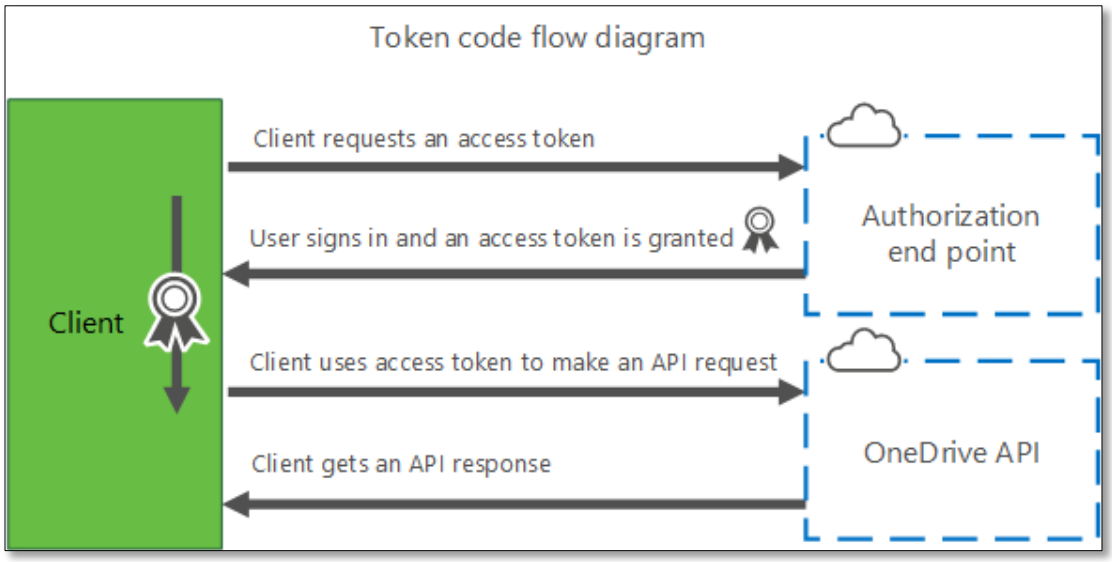


Abbildung 13: Das Token-Verfahren [11]

OAuth bietet zwei Verfahren an: Den Token-Flow und den Code-Flow.

Im Token-Verfahren (Abb. 13) muss der Nutzer sich zunächst auf dem Client mit den Login-Daten einloggen. Der Nutzer stimmt dabei vorgezeigten Rechten, wie bspw. dem Anzeigen der E-Mail-Adresse und weiteren Profilinformationen zu. Das Login-Fenster basiert zwar auf der Anwendung, der Client schickt die Daten jedoch ohne Verarbeitung an den Authorization Server. Als Alternative könnte der Client den User zur zugehörigen Seite weiterleiten. Über einen Redirect kommt der Nutzer nach dem Login zurück in die Anwendung.

Der Authorization Server validiert die Daten. Bei Gültigkeit generiert er einen Zugangsschlüssel, den Access Token. Diesen kann der Client dann für direkte Autorisierungen mit dem entfernten Server einsetzen. Im Access Token liest der Server zum einen den Benutzer raus und zum anderen welche Rechte der Client von ihm bekommen hat. Ohne den Access Token ist es also nicht möglich, Aktionen durchzuführen. Access Token haben dabei eine beschränkte Gültigkeit. Sie müssen in den meisten Fällen nach wenigen Stunden erneuert werden.

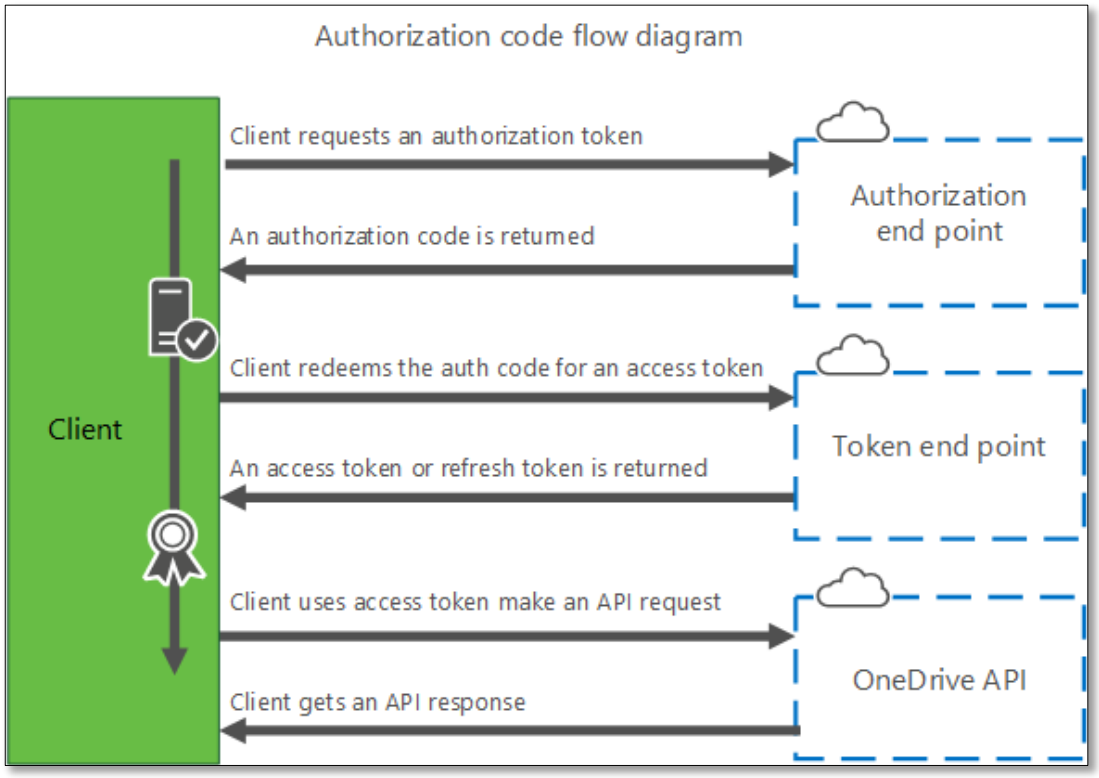


Abbildung 14: Das Code-Verfahren

Das Code-Verfahren (Abb. 14) benötigt einen weiteren Schritt. Zunächst wird ein Authorization Token durch Authentifizierung des Clients angefragt. Der Client muss dem Server seine ID und sein Secret (Passwort) senden. Beides wird durch die Anmeldung der Anwendung auf der Website generiert. Der Server definiert einen Code und schickt ihn dem Client. Der Client kann diesen Code nun gegen einen Access Token einlösen, indem sich der Nutzer authentifiziert. Mit dem Access Token kann der Client nun wieder die API Calls durchführen.

Im Gegensatz zum Token-Verfahren, findet beim Code-Verfahren eine doppelte Überprüfung statt. Zum einen authentifiziert sich der Nutzer der Anwendung, zum anderen die Anwendung selbst. Anwendung und Nutzer sind also beide dem Server bekannt. Beim vorher gezeigten Token-Verfahren weiß also die API nicht genau, wer die Requests im Namen des Nutzers ausführt. Das Token-Verfahren findet daher vor allem bei Geräten Anwendung, bei denen die ID und das Passwort geheim bleiben sollten. Das Code-Verfahren wird verstärkt bei Web-APIs genutzt, um Angriffe zu vermeiden. Facebook, Spotify und Nest sind Beispiele für APIs, die nur durch das Anlegen einer Anwendung auf der Website funktionieren. Andere APIs wie OneDrive bieten beide Verfahren an.

1.8.9.2 HTTP Basic Authentication

Eine weitere Möglichkeit der Authentifizierung ist die HTTP Basic Authentication. Bei diesem einfachen Schema verwendet der Client einen Benutzernamen und ein Passwort, trennt sie durch einen Doppelpunkt und codiert sich nach dem Base64[23]-Verfahren. [52] Das Ergebnis wird im Header unter dem Parameter Authorization eingetragen. Der Server kann die Codierung lösen und die Werte auslesen. Der Vorteil ist die einfache Implementierung.

Während bei OAuth die Kommunikation über Access Tokens verläuft, die keine Passwörter des Nutzers enthalten, muss bei der Basic Authentication Nutzername und Passwort bei jedem API Call mitgeschickt werden. Eine zusätzliche SSL-Codierung durch HTTPS ist demnach empfehlenswert.

1.8.10. Die Facebook API

Mit der Facebook API lassen sich alle Funktionen von Facebook extern nutzen. Für die Nutzung der API ist ein Benutzerkonto auf Facebook Voraussetzung. Facebook stellt für Entwickler Javascript SDKs zur Verfügung.

Client und Server

Der Client ist die Anwendung des externen Entwicklers. Er ist der PC oder Server, auf dem die Anwendung verarbeitet wird. Damit ist also nicht der Nutzer gemeint, der die Anwendung ausführt. Den Server stellt die URL `https://graph.facebook.com` dar. An diese können die Requests geschickt und analog Responses empfangen werden.

Authentifizierung

Zunächst muss geklärt sein, welche Funktionen die Anwendung bekommen soll. Ein Beispiel ist der Login, der die Voraussetzung für weitere Aktivitäten ist. Diese können das Posten von Inhalten auf der eigenen Seite, das Einsehen der Gefällt-mir-Angaben oder das Ansehen der eigenen Zeitleiste beinhalten. Dann können die Permissions[24] der App festgelegt werden. Für den Login muss der Nutzer einwilligen, dass sein öffentliches Profil (*public profile*) benutzt wird. Für die Gefällt-mir-Angaben benötigt Facebook die *user likes*.

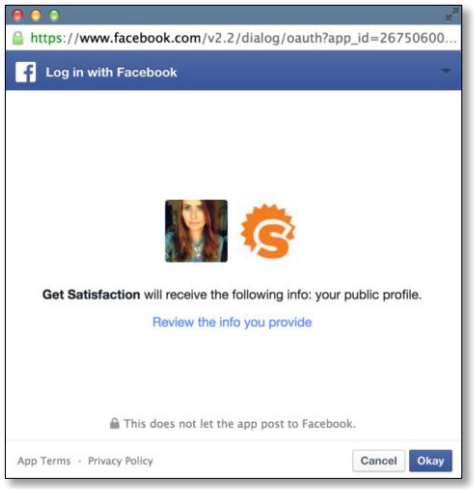


Abbildung 15: Berechtigungsabfrage beim Login [12]

Diese Berechtigungen werden beim Login des Nutzers abgefragt (siehe Abb. 15). Die Verifizierung verläuft über das im vorherigen Kapitel vorgestellte OAuth. Facebook nutzt das Code-Verfahren, d.h. es müssen eine App-ID und App-Secret übergeben werden.

[23] Base64 ist ein Verfahren, bei dem die gegebene Zeichenkette in Buchstaben codiert wird. So können beispielsweise Binärdateien als einfacher Text angezeigt werden.
[24] dt.: Berechtigungen

Beides bekommt der Entwickler durch das Erstellen einer App auf Facebook. Diese App beinhaltet Informationen zu der Anwendung, die der Entwickler schreibt. Facebook generiert daraufhin für die App eine ID und ein Passwort. Mit diesen kann der Entwickler nun die Authentifizierung seiner Anwendung durchführen.

Für die Nutzer-Authentifizierung stellt die externe Anwendung einen Login-Dialog zur Verfügung oder leitet den Nutzer direkt an Facebook weiter. Die Anwendung verarbeitet dabei zu keinem Zeitpunkt die Daten.

Im nächsten Schritt muss der Access Token angefragt werden. Nachdem der Nutzer sich über den Dialog bei Facebook angemeldet hat, generiert Facebook einen Access Token, mit dem die durch die Berechtigungen zugelassenen Funktionen durchgeführt werden können und der Nutzer identifiziert wird.

Beispielhafte Requests und Responses

Voraussetzung für das Posten von Inhalten ist die Permission *publish actions*. Über den Pfad *me* können nun Inhalte im Namen des Nutzers veröffentlicht werden:

```
POST v2.5/me/feed?message=Mein+erster+Post! HTTP/1.1
Host: https://graph.facebook.com/
```

Durch den POST-Befehl wird ein Wert hinzugefügt. Danach folgt der Befehl: *v2.5* steht für die Versionsnummer des Graph-Servers, *me* für die eigene User-ID und *feed* für die Timeline[25]. Mit dem Query *message*=.. können beliebige Nachrichten übergeben werden.

Als Antwort schickt Facebook die User-ID gefolgt von der Post-ID. Antworten erfolgen üblicherweise in JSON-Format:

```
{
  "id":"NUTZER-ID_POST-ID"
}
```

Die Gefällt-mir-Angaben können durch einen GET-Request abgerufen werden:

```
GET v2.5/me/likes     HTTP/1.1
Host: https://graph.facebook.com/
```

Facebook sendet daraufhin alle Seiten, die dem Nutzer gefallen. Die Daten beinhalten den Namen der Seite, Seiten-ID und Zeitpunkt des Likes:

```
{
  "data": [
    {
      "name": "20th Century Fox",
      "id": "172660165927",
      "created_time": "2015-05-17T18:34:09+0000"
    },
    {
      "name": "Hochschule Offenburg",
      "id": "154796461228006",
      "created_time": "2013-07-22T10:38:36+0000"
    }
}
```

[25] dt. Zeitleiste; steht in Facebook für den öffentlichen Nachrichtenverlauf

Social Plug-ins

Social Plug-ins, wie der Like- oder Share-Button, können über SDKs implementiert werden. In diesen vorgegebenen Codes lassen sich die fokussierte Seite und Darstellung der Buttons auswählen.

```
<div class="fb-like"
       data-href="http://www.facebook.com/CloudRail-Dashboard-
966000033460636/.html"
       data-layout="standard"
       data-action="like"
       data-show-faces="false">
</div>
```

In die *data-href* wird die Ziel-Domain geschrieben, die der Like referenziert. Im Beispiel-Code ist die Facebook-Seite für das im Projekt erstellten Dashboard verlinkt. Durch *data-layout* kann unter den oben dargestellten Layouts ausgewählt werden. Die *data-action* beschreibt die durchzuführende Aktion, in diesem Fall der Like. Mit *data-show-faces* kann definiert werden, ob unter dem Like-Button das Profilfoto des Nutzers angezeigt werden soll.

Abbildung 16: Facebook-Like [13]

Abbildung 16 zeigt die verschiedenen Style-Möglichkeiten. Der Entwickler kann die Funktionen und Designs als Framework in seinen Code implementieren.

Kapitel 2 CloudRail

Unter dem Namen licobo GmbH gründeten Felix Kollmar und David Amann 2012 ein Unternehmen, mit dem sie ein sicheres Online-Adressbuch in die Welt bringen wollten. Im Jahr 2014 verwarfen sie das Vorhaben und hatten eine neue Idee. Mit dem aufkommenden Trend des IoE (Internet of Everything) entwickelten sie eine universelle API. Mit ihr sollen Entwickler im IoE unterstützt werden. Abbildung 17 zeigt das Firmenlogo.
Es soll visualisieren, dass durch die CloudRail-Lösung mehrere Clouds miteinander vernetzt werden können.

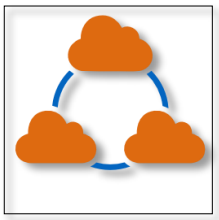


Abbildung 17: Das CloudRail-Logo [14]

„Our technology is a major milestone in the evolution of the Internet of Everything and enables the dream of a connected world to become true." [55]

Die Motivation von CloudRail ist klar: Das IoE hat das Potential, die Welt zu verändern. Momentan gibt es wenige Lösungen für die Vernetzung von mehreren Geräten und Diensten. Dadurch dass jedes einzelne Gerät über eine Schnittstelle agiert, müssen Unternehmen einen Plan entwickeln, wie diese Vernetzung leichter und schneller passieren kann. CloudRail bietet einen solchen Plan.

2.1. Fakten

Gegründet	2012 als licobo GmbH
Geschäftsführer	Felix Kollmar, CEO
	David Amann, CTO
Mitarbeiter	9 (Stand: Januar 2015)
Sitz	Julius-Hatyr-Str. 1
	68163 Mannheim
Branchenschwerpunkte	IT-Development
	Dienstleistung

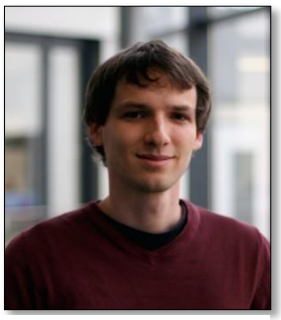


Abbildung 18: CEO Felix Kollmar [15]

2.2. Über CloudRail

CloudRail ist ein Start-Up. Dies bedeutet, dass die Firmengründung nicht nur auf Eigeninvestitionen basiert, sondern auch externe Förderer beteiligt sind. Für ihr Vorhaben konnte CloudRail etwa eine halbe Million Euro sammeln, welches von Leonardo Venture[26], dem Beteiligungsfonds Wirtschaftsförderung Mannheim[27] und mehreren Business-Angels zur Verfügung gestellt wurde.

Seit dem 05.11.2015 ist die API auf dem Markt. Vorher konnten Entwickler lediglich eine Beta-Version nutzen. Nun ist die API aber mit vollem Support erhältlich.

2.3. Release

Die Workbench, auf der APIs definiert und eigene Endpunkte erstellt werden können, kann seit Release online unter `http://workbench.cloudrail.com` aufgerufen werden. Die Abbildung 19 zeigt die offizielle Pressemitteilung zum Release.

[26] eine Venture-Capital-Gesellschaft (Risiko-Kapital) mit Fokus auf Unternehmen in einer frühen Entwicklungsphase (Early-Stage); http://leonardoventure.com/
[27] https://www.mannheim.de/wirtschaft-entwickeln/beteiligungsfonds-wirtschaftsfoerderung-mannheim

Für sofortige Veröffentlichung: November 5, 2015

Felix Kollmar
CloudRail
+49 621 48345965
press@cloudrail.com

CLOUDRAIL – die "UNIVERSELLE API FÜR ALLES"

Eine einzige Schnittstelle für die Integration jeglicher Cloud-Dienste oder IoT-Geräte

Mannheim – CloudRail, ein Visionär der vernetzten Welt, hat heute die "Universelle Schnittstelle für Alles" auf den Markt gebracht. Die neue Lösung ermöglicht es Software Entwicklern, ihre Anwendung über eine einzige gemeinsame Schnittstelle mit einer Vielzahl von Cloud-Diensten und intelligenten Geräten zu verbinden.

APIs (Application Programming Interfaces) versetzen Anwendungen und Dienste oder Geräte in die Lage, miteinander zu kommunizieren. Sie ermöglichen nahtlose Vernetzung und schaffen somit letztlich die Basis für das „Internet of Everything". Entwickler stehen heute aber noch vor der Herausforderung, dass jeder Dienst und jedes Gerät eine eigene API mit herstellerspezifischem Datenformat und eigener Semantik nutzt. Jede einzelne Integration muss deshalb manuell vorgenommen werden und ist zeitaufwändig. Auf der anderen Seite stehen Cloud-Anbieter vor der Herausforderung ihre Dienste auf einfache Art mit möglichst vielen Apps kompatibel zu halten.

CloudRail löst dieses Problem. Softwareentwickler können CloudRail für die automatische Integration in ein breitgefächertes Angebot führender Cloud-Dienste und intelligenter Geräte nutzen. Das mit dem Launch zur Verfügung stehende Angebot beinhaltet Integrationen für die wichtigsten Dienste, darunter Dropbox und Facebook, sowie intelligente „Smart Home"-Geräte wie Nest und Philips Hue. Zusätzlich können Entwickler über die vorhandenen Integrationen hinaus weitere Funktionalität oder komplett neue Dienste auf einfache Weise zur CloudRail Integrations-Datenbank hinzufügen. Im Ergebnis ermöglicht die Universelle CloudRail API eine starke Verringerung der Entwicklungszeit und damit deutlich mehr kompatible Apps und Geräte.

Die Universelle CloudRail API steht kostenfrei für Softwareentwickler zur Verfügung. CloudRail und die Entwickler-Community schaffen so gemeinsam die Standard-Plattform für das interoperable Internet von morgen.

Entwickler können den Dienst sofort unter http://cloudrail.com nutzen. CloudRail steht aktuell für Java und Android zur Verfügung. IOS und weitere Plattformen werden in Kürze folgen.

Über CloudRail
CloudRail's Vision ist eine einzige, universelle Schnittstelle für die vernetzte Welt von morgen. Die universelle CloudRail API ermöglicht es Entwicklern Cloud-Dienste oder intelligente Geräte schneller, günstiger und sicherer einzubinden, als das mit bisherigen Mitteln möglich wäre. Mit Hauptsitz in Mannheim und einer Niederlassung im kalifornischen San Francisco, ermöglicht CloudRail die Realisierung eines interoperablen Internets. Erfahren Sie mehr auf http://cloudrail.com

Abbildung 19: Die Pressemitteilung zum Release

2.4. Das Interoperabilitätsproblem

Interoperabilität ist die Kernkompetenz, die im IoE gewährleistet sein muss. Der Begriff beschreibt die Fähigkeit einzelner Systeme zur Kommunikation mit weiteren. Darunter versteht sich das Austauschen von Daten ohne die Erforderlichkeit, gesonderte Vereinbarungen unter den Systemen zu treffen.

Die Vernetzung verschiedener Geräte bedeutet die Zusammenführung verschiedener Schnittstellen, Datenformaten und der Semantik. Dies führt dazu, dass die Geräte nicht interoperabel sind oder einfacher ausgedrückt, sie sprechen verschiedene Sprachen und können sich nicht problemlos verständigen. Das Interoperabilitätsproblem beschäftigt seit Jahren Firmen aus dem IT-Sektor.

2.4.1. Standards im IoE

Eine Lösung für das Problem ist ein gemeinsamer Standard unter den Herstellern. So können die Geräte in der gleichen Sprache kommunizieren und vernetzt werden. In den letzten Jahren haben sich Herstellerverbünde gebildet, die zusammen Lösungen schaffen wollen. Einige dieser Verbünde werden folgend mit ihren Zielen vorgestellt:

- **Industrial Internet Consortium (IIC)**
  Seit 2014 gibt es ein Konsortium mehrerer IT-Firmen mit Fokus auf die Industrie. Zu dem Non-Profit-Verbund IIC können und sollen Forschungsinstitute und öffentliche Einrichtungen beitreten. Als Ziele setzt sich das Konsortium, Innovationen für die Anwendung in der realen Welt durch Use-Cases und Testumfelder voranzutreiben, Frameworks und Referenzarchitekturen für Interoperabilität zu entwickeln und damit den globalen Standard-Entwicklungsprozess zu beeinflussen. [56]

- **Open Interconnect Consortium (OIC)**
  In Erweiterung zum IIC will das OIC einen offenen Standard, der für alle von Nutzen ist. Die Industrie, aber auch Konsumenten und der Gesundheitssektor sollen profitieren. Der Standard soll als Framework realisiert werden und einfach zu implementieren sein. Für das Framework spezifiziert das OIC eine Vielzahl an neuen Kommunikationsmethoden, Identitätssicherung und On-Boarding-Support. Zwischen August und Oktober 2015 stellte das OIC Mitgliedern eine Vorschau der Spezifikationen zur Verfügung. Wann das Framework für die Öffentlichkeit zugänglich wird, ist unklar. [57]

- **AllSeen-Alliance**
  AllJoyn von Qualcomms ist ein P2P-Framework, durch das Anwendungen erstellt werden können, die eine Nahfeldkommunikation ermöglichen. Hierzu muss kein Server als Vermittler eingebunden werden. [58] Die AllSeen-Alliance, zu der Qualcomms zählt, stellt die plattformunabhängige Technik als Open-Source zur Verfügung.

- **Thread**
  Thread konzentriert sich auf den SmartHome-Bereich. Demnach sind vorwiegend Firmen der Branche, wie Yale, Samsung und Silicon Labs im Verbund vertreten. Thread will ein Netzwerk-Protokoll entwickeln, das eine große Bandbreite an Produkten unterstützt. Ziele sind neben der Vernetzung eine einfache Nutzung, Sicherheit und Energieeffizienz. [59]

- **UPnP**
  UPnP beinhaltet mehrere Netzwerk-Protokolle, die vor allem dafür eingesetzt werden, andere Geräte zu lokalisieren und eine Verbindung über ein bekanntes Netzwerk aufzubauen. Die Protokolle werden von Foren-Mitgliedern, zu denen einige führende Unternehmen aus der Computerindustrie zählen, gemeinsam definiert. UPnP basiert auf Protokoll-Standards wie HTTP, UDP, TCP und IP. [51] Es eignet sich aber lediglich für den Multimedia-Bereich. Denkbare Vernetzungen sind das Teilen von Informationen unter Geräten und dem World Wide Web, das Verschieben von digitalen Daten wie Fotos, Videos und Musik zwischen Geräten, sowie das Fernsteuern von Geräten. [60]

- **DLNA**
  Auch DLNA ist ein Herstellerverbund, der Protokolle und Standards zur Vernetzung multimedialer Geräte entwickelt. Der Standard sorgt dafür, dass die Formate der Bilder, Videos und Musik von allen Geräten unterstützt werden. Über einen DLNA-Server, der bei vielen Smartphones vorinstalliert ist, oder per WiFi Direct können die Medieninhalte ausgetauscht werden. [61]

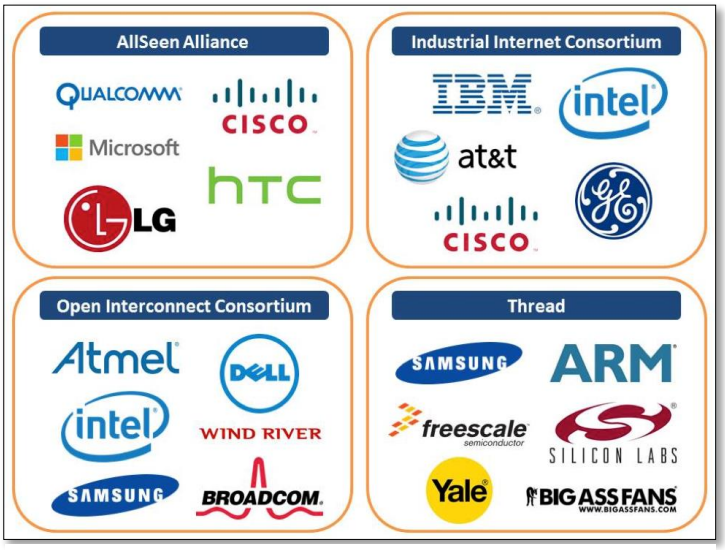


Abbildung 20: Die Verbünde und ihre Mitglieder [16]

Entsprechend den Zielen des Verbundes, finden sich die Hersteller des fokussierten Sektors zusammen (siehe Abb. 20). So sind beim IIC industrielle Dienstleistungsunternehmen wie IBM oder Cisco und bei Thread privatkonsumorientierte Unternehmen wie Samsung vertreten.

Zwar können Standards kurzfristig das Interoperabilitätsproblem lösen, indem sich die Hersteller auf bestimmte Regeln festlegen, mit der zunehmenden Zahl an Geräten und Möglichkeiten werden Standardisierungsprozesse im IoE aber entweder Jahre dauern oder erfolglos abgebrochen. Denn um flexibel zu bleiben, schwächen Firmen die Standards so stark ab, dass sie nicht mehr kompatibel sind.

Allgemein bieten Standards keine optimale Lösung, da der Entwicklungsprozess im IoE unter dem Standard eingeschränkt werden würde. Dies ist in einem so neuen Gebiet ein denkbar schlechtes Szenario. Ein letzter Grund gegen Standards sind die Firmen selbst. Ein Standard hat für Entwickler die Bedeutung, dass beispielsweise Apps aus dem Apple Store auch auf Android Geräten funktionieren müssen. Die Idee vieler Firmen ist aber, dass sobald sich der Kunde für ein System entschieden hat, er auch nur deren Geräte, Produkte und Dienstleistungen in Anspruch nimmt. Dieses Konzept ist unter dem Begriff *Walled Garden*-Problem bekannt geworden. [62]

2.5. Die CloudRail Lösung

Um Geräte standardlos in das Internet der Dinge zu integrieren, bietet CloudRail eine universelle API an. Mit der API wird der Entwicklungsprozess von Anwendungen vereinfacht und darüber hinaus ein Grundstein der geräteunabhängigen Vernetzung gelegt.

Abbildung 21 auf der folgenden Seite zeigt wie ein Entwickler ohne CloudRail verschiedene Dienste in seine Anwendung integrieren muss.

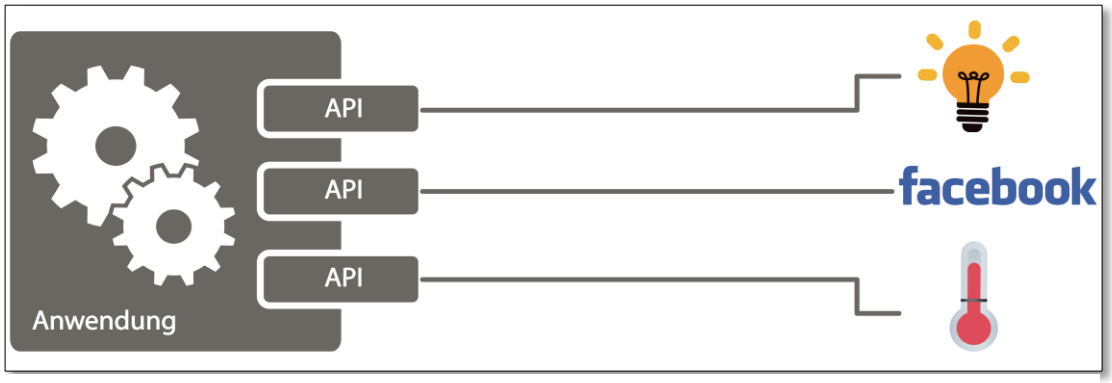


Abbildung 21: APIs im Internet der Dinge

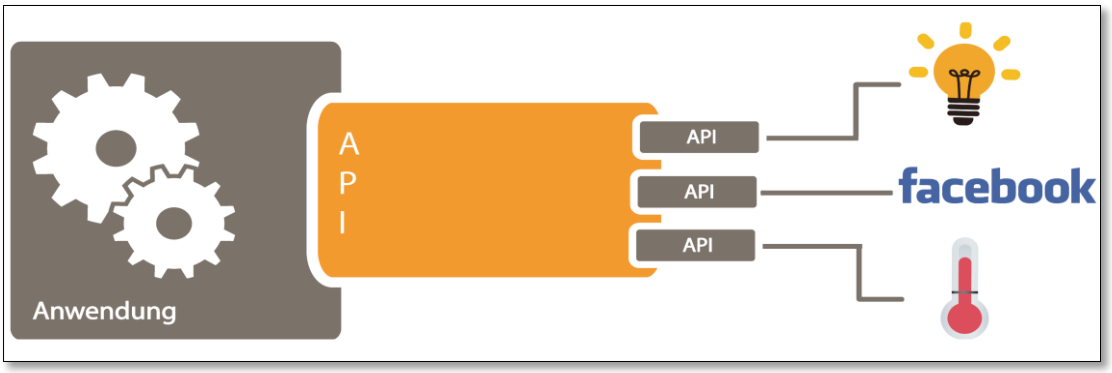


Abbildung 22: Die CloudRail-API

Jeder Dienste, hier visualisiert durch die Philips Hue Lampe, Facebook und das Nest Thermostat, findet über eine eigene API Platz in der Anwendung. Der Entwickler muss dort mit verschiedenen Datenformaten, unterschiedlicher Semantik der Daten und vor allem der Vielzahl an APIs umgehen. Die Kommunikation zwischen den Geräten ist auf diese Weise nicht möglich. Besser wäre es, die Geräte nur über eine universelle API in die Anwendung zu integrieren.

Um das Problem der unterschiedlichen Datenformate zu lösen und den Entwickler in der Programmierung seiner Anwendung zu unterstützen, übernimmt CloudRail die Konvertierung der verschiedenen Formate und stellt dem Entwickler darüber hinaus ein SDK zur Verfügung. Während ohne CloudRail das Problem der Interoperabilität in der Anwendung gelöst werden muss, bietet CloudRail eine umfassende Lösung dafür und macht das Erstellen von Anwendungen im Internet der Dinge für jeden zugänglich. In der CloudRail-Schnittstelle können Geräte mit unterschiedlichen Sprachen definiert und damit im Internet der Dinge vernetzt werden.

Die folgenden Kapitel zeigen den Aufbau der CloudRail-Workbench und wie die SDK den Entwickler in der Anwendung unterstützt.

2.5.1. Die CloudRail-Workbench

Durch die CloudRail-Workbench können eigene Endpunkte entwickelt werden. Der Endpunkt beinhaltet alle vernetzten Geräte und verknüpfte Methoden. Hierzu gibt es die Möglichkeit, bereits vorhandene API-Definitionen zu nutzen oder weitere anzulegen. Für die Nutzung der CloudRail-Workbench ist ein Account Voraussetzung.

Folgend wird ein Überblick über die Workbench gegeben.

Startseite
Die Abbildung 23 zeigt die Startseite der Workbench. Diese Seite wird nach dem Login auf der Workbench aufgerufen.

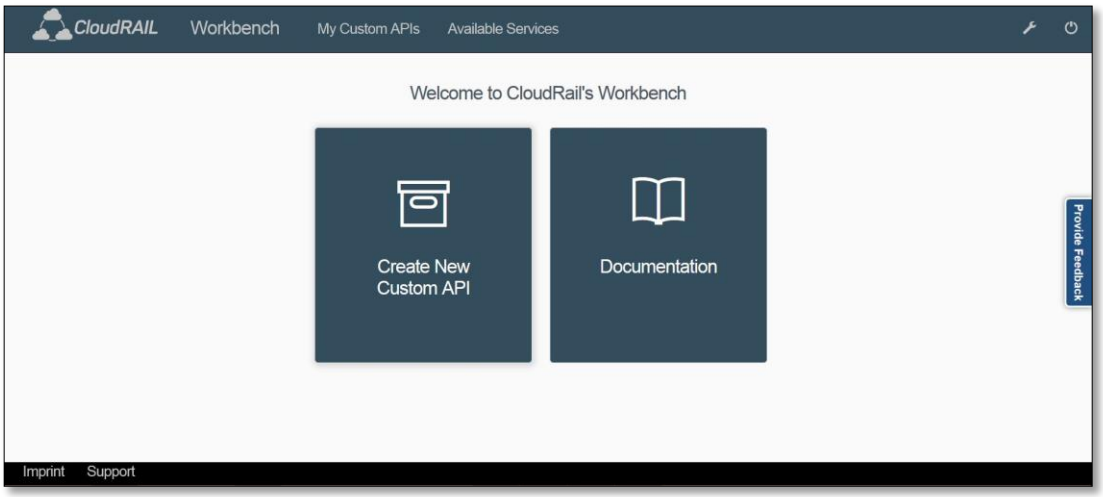


Abbildung 23: Die Startseite der CloudRail-Workbench [17]

Auf der Startseite kann direkt zu einem eigenen Projekt, verfügbaren Services oder einer Schnellstart-Anleitung navigiert werden. Dort wird einfach und verständlich beschrieben, wie die erste Schnittstelle erstellt werden kann.

Verfügbare Dienste

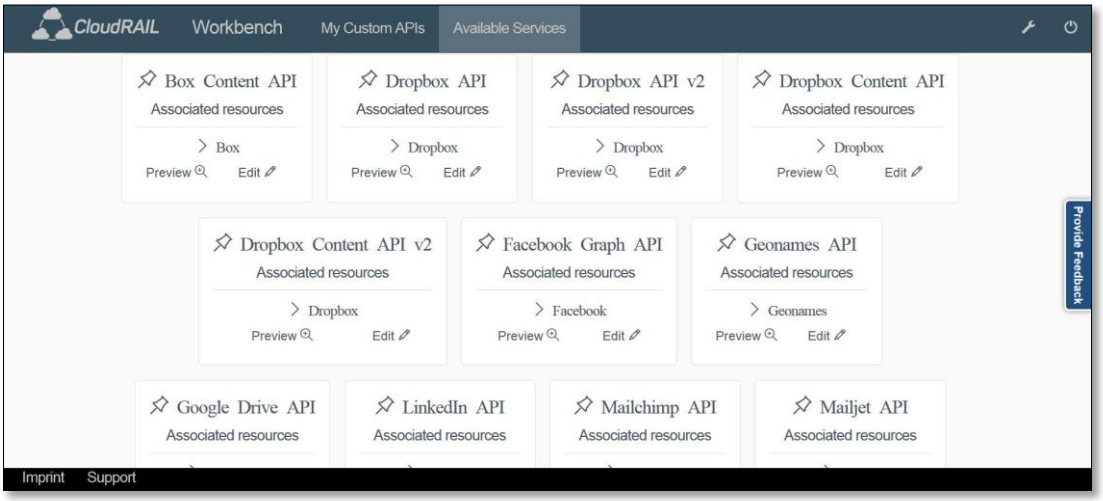


Abbildung 24: Verfügbare Dienste [17]

CloudRail stellt einige definierte APIs zur Verfügung. Abbildung 24 zeigt die Seite mit einer Auswahl. Entwicklern steht es frei, diese APIs nahezu ohne Aufwand in ihre Anwendungen zu integrieren. Fehlende APIs können auf einfache Weise hinzugefügt werden. Die Verfahren der Definierung von APIs variiert zwischen Herstellern, da sich die APIs grundlegend in Request und Response, Autorisierung, Methoden und Variablen unterscheiden.

In den ersten Monaten wurden eine Vielzahl an API-Definitionen hinzugefügt. Folgend ist eine kategorisierte Liste mit den zur Verfügung stehenden APIs dargestellt (Stand: Januar 2016):

- **Nachrichtentransfer**
  Twilio, Mailchimp, Mailjet, Sendgrid, Nexmo, Slack
- **Musik**
  Spotify, Soundcloud
- **Smart Home**
  Nest Thermostat, Netatmo, Philips Hue
- **Sozial**
  Facebook, LinkedIn, Google+
- **Storage**
  Dropbox, Google Drive, Box.com
- **Wetter**
  OpenWeather, AccuWeather
- **Kartierung**
  Geonames, Google Places
- **Finanzen**
  Stripe, Venmo
- **Sonstig**
  Uber, YouTube, Jawbone

Meine APIs

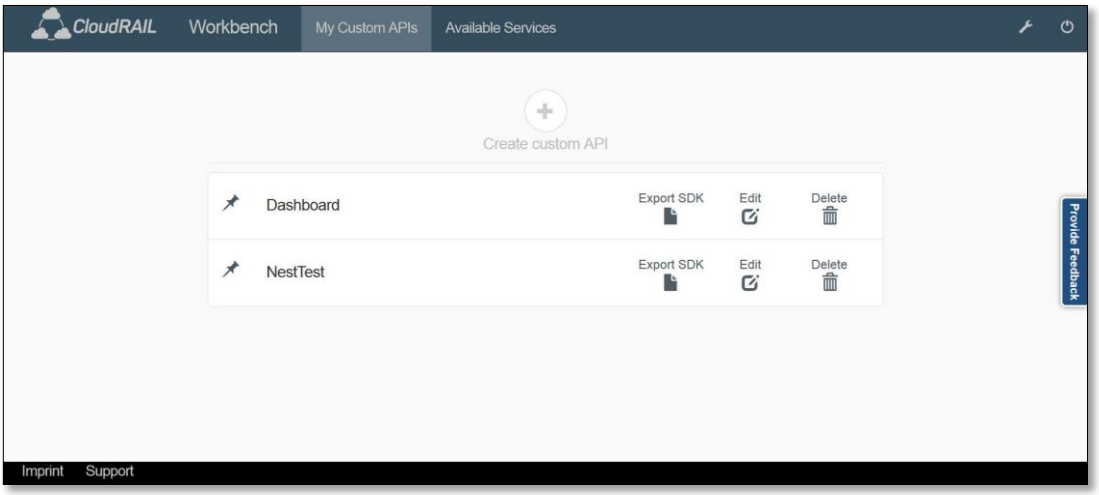


Abbildung 25: Meine APIs [17]

Zuletzt erfolgt die Erstellung des Endpunktes. Abbildung 25 zeigt den Überblick der Projektseite. Das Dashboard-Projekt wurde bereits angelegt. Dieses kann nun weiter konfiguriert werden oder als Java-SDK und Konfigurationsdatei auf den PC heruntergeladen werden. Jegliche Fortschritte werden auf der Workbench gespeichert, sodass der Nutzer beim erneuten Einloggen die Einstellungen wiederaufgreifen kann.

2.5.2. Funktionen der Workbench

In der Workbench können also APIs und Endpunkte definiert werden. Ein Endpunkt steht für ein Projekt, in dem mehrere API-Aktionen ausgeführt werden können. Dies wurde im vorherigen Kapitel mit dem Dashboard-Projekt beispielhaft gezeigt.

Die API-Definitionen führen im Hintergrund zu einem Graph mit allgemeinen Informationen. Graphen ermöglichen die plattformunspezifische Definition von Zusammenhängen. Durch sie können kürzere, effiziente Wege zum Ziel gefunden werden. Die Informationen des ersten Graphen beinhalten Protokolle, URLs und globale Variablen. Die Funktionen der API, beispielsweise das Abfragen von Wetterdaten oder Teilen von Inhalten auf Facebook führen zu weiteren Graphen. Diese beinhalten die jeweiligen Operationen.

Auch aus der Endpunkt-Definition lassen sich Graphen ableiten. Der erste Graph beinhaltet Informationen zur Autorisierung, wie die Autorisierungs- und Redirect-URL. Er ist des Weiteren mit den Aktionen verlinkt und führt alle globalen Platzhalter seiner Funktionen auf. Ein weiterer stellt die verschiedenen Aktionen (Actions) dar. Er zeigt, welche Actions zu welchen Operationen führen. Da mehrere API-Calls durch eine Action möglich sein sollen, ist die Trennung von Operationen und Actions wichtig. Kurz gesagt liegt der Unterschied darin, dass Actions keine optionalen Parameter abbilden, sondern nur jene, deren Angabe auf jeden Fall benötigt wird. Operationen hingegen weisen alle Parameter auf. Die Endpunkt-Definition ist letztlich wichtig, um den Input des Nutzers zu verarbeiten. Aus den API- und Endpunkt-Definitionen entsteht die Graph-Datei, die Entwickler in ihr Projekt einbinden können. Die Graph-Datei wird gemeinsam mit dem SDK heruntergeladen.

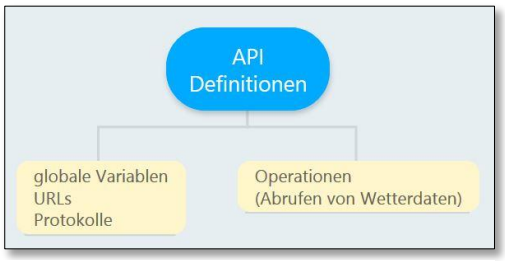


Abbildung 26: Graphen der API-Definitionen

Abbildung 27: Graphen der Endpunkt-Definitionen

Zusammenfassend führen auf der Workbench erstellte Definitionen der APIs und Endpunkte also zu Graphen. Im Fall der APIs entstehen Graphen mit Informationen über die Operationen und globale Variablen (siehe Abb. 26). Im Fall des Endpunktes entstehen Graphen mit Verknüpfungen zu den Operationen, sowie Informationen über globale Variablen und Autorisierungs-URLs (siehe Abb. 27).

2.5.3. Das CloudRail-SDK

Um die SDK begreiflicher zu machen, wird der Workflow vom Input des Nutzers bis zur Ausgabe des Ergebnisses folgend dargestellt.

Der Nutzer schreibt Daten in die Schnittstelle. Dies kann durch die Eingabe von Text in ein Textfeld, einen Buttondruck oder andere Werteeingaben passieren. Globale Variablen, wie IP des Nutzers, ID des Clients oder Access Token werden an den Graph der Endpunkt-Definition angehängt. Dies hat den Hintergrund, dass dies nur einmal bei wiederholenden Anfragen erfolgen muss. Lokale Variablen und die Eingaben des Nutzers werden wiederholend an den Graph der Actions angehängt.

Die Engine von CloudRail führt nun die Verarbeitung durch. Sie schaut, ob sie entweder die Graph-Operationen auf bekannte Äquivalenzknoten abbilden kann oder ein festgelegtes Muster in der Graphstruktur erkennt, um spezifische Plug-ins zu aktivieren. Die Engine arbeitet solange, bis sie keine Übereinstimmungen mehr findet.

Die Graph-Actions werden folgend mit den Operationen abgestimmt und die Inputs werden ausgelesen. Auf ein bestimmtes Muster in den Operationen hin fängt das zuständige Plug-in an zu arbeiten. Der Vorteil dieses Ablaufs liegt darin, dass die Graphen plattformunspezifisch sind und lediglich die Plug-ins in verschiedene Sprachen übersetzt werden müssen. Durch die Erkennung von Mustern im Graph wird das Plug-in ausgewählt. Die Plug-ins übertragen nun die Daten in einem HTTP Request an die externe API, beispielsweise die Facebook API und warten eine Antwort ab.

Die Antwort, die wie beschrieben meistens in XML oder JSON erfolgt, wird extrahiert und an den Graph-Operationen angehängt. Diese werden wieder auf die Actions übertragen. Die Engine ist an diesem Zeitpunkt fertig, sofern keine weiteren Daten eintreffen. Sie merkt sich die gerade durchgeführte Paarung, um den Vorgang nicht unnötigerweise zu wiederholen. Die Schnittstelle ruft nun die Daten ab und gibt sie an den Nutzer aus.

In Abbildung 28 ist eine Zusammenfassung des Ablaufs vereinfacht skizziert. Im Mittelpunkt steht die Schnittstelle, die die Daten des Nutzers verarbeitet. Lokale Variablen werden wiederholend an den Graph der Actions gehängt. Globale Variablen werden bei Instanziierung der SDK einmalig übergeben. Die Engine ist nun für die Verarbeitung des Request zuständig. Actions werden auf Operationen abgebildet. Auf Muster in den Operation-Graphen reagieren Plug-ins, die im direkten Austausch mit der externen API stehen. Sie senden also den HTTP-Request und erwarten die Response. Die Response wird extrahiert und an den Operations-Graphen angehängt (roter Pfeil). Die Operationen werden daraufhin auf Actions übertragen (orangener Pfeil). Schließlich ruft die Schnittstelle die Daten aus den Actions ab (grüner Pfeil) und gibt sie an den Nutzer aus (schwarzer Pfeil).

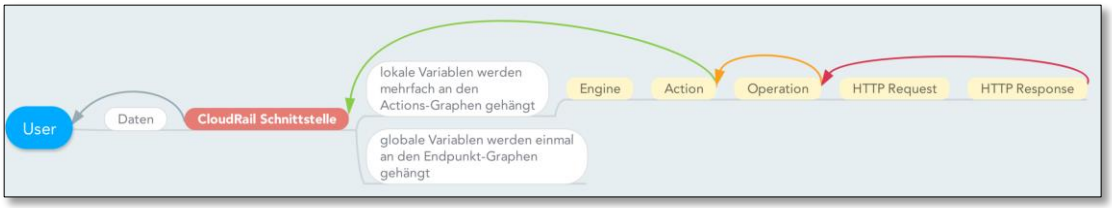


Abbildung 28: Zusammenfassung der Funktionsweise

2.5.4. Workflow des Entwicklers

Durch die vordefinierten Interfaces in dem SDK kann der Entwickler seine Input-Felder mit den jeweiligen Methoden verbinden. In diesem Fall ist der Name der SDK-Klasse Dashboard. Die zugehörige Variable wird kleingeschrieben.

```
citySubmit.addClickListener(new ClickListener() {

      @Override
      public void buttonClick(ClickEvent event) {

      dashboard.getWeatherByCity(cityInput.getValue(), apiKey, new
      GetWeatherByCityResponseListener() {

            public void onSuccess(String cityname, Double temperature,
            Long humidity, Long cloudiness) {

            String outcome = "My weather data for " + cityname + ": The
            thermostat shows " + temperature" °C, humidity is about " +
            humidity + "% and the sky is filled with "+ cloudiness + "%
            clouds.

      public void onError(CloudRailException error) {}
      public void onProgress(double percentFinished) {}
      });
}});
```

Code 1: Integration der SDK in den eigenen Code

Der Codeausschnitt 1 zeigt die Verbindung des Codes mit dem SDK. Um mit der ersten Zeile zu starten, wird dem Button *citySubmit* ein *ClickListener* hinzugefügt. Dieser führt zu Aktionen (*ClickEvents*), sobald der Nutzer den Button klickt. Ist dies der Fall, wird die Methode *getDataByCityName* der Dashboard-Klasse aufgerufen. Diese wurde auf der Workbench mit ihren Inputs und Outputs als Endpunkt-Definition festgelegt.

Globale Variablen müssen bereits bei der Instanziierung der *dashboard*-Variablen übergeben werden. Im Fall von OpenWeather gibt es keine, da für die Benutzung der API keine Autorisierung benötigt wird. An die Methode werden nun der vom Nutzer eingetragene Name der Stadt, der API-Key und ein *Response-Listener* übergeben.

Der API-Key wird durch Anmeldung bei OpenWeather generiert und verliert nicht an Gültigkeit. Der *ResponseListener* übergibt drei Methoden: *onSuccess*, *onProgress* und *onError*. In *onSuccess* können Aktionen bei erfolgreichem API-Call definiert werden. Die Methode liefert die festgelegten Outputs als Parameter. Im Code-Beispiel werden die Variablen im String *outcome* gespeichert und in der Konsole ausgegeben. Eine beispielhafte Ausgabe für die Stadt Mannheim wäre demnach:

```
My weather data for Mannheim: the thermostat shows 10.32°C,
humidity is about 81% and the sky is filled with 90% of
clouds.
```

onError beinhaltet Aktionen bei fehlerhaftem Durchlauf. In *onProgress* kann der Fortschritt des Calls ausgegeben oder visualisiert werden.

Mit diesem Beispiel wurde gezeigt, dass der Entwickler letztlich keinen Kontakt mit dem Endpunkt aufnehmen muss. Die entsprechenden Calls werden von vordefinierten Methoden ausgeführt, die aus dem generierten SDK importiert werden können. Zu den Aufgaben des Entwicklers zählen also lediglich die Implementierung seiner Anwendung und die Integration der CloudRail-SDK. Durch diese kann er sich vollständig auf die Entwicklung jener Anwendung fokussieren.

Kapitel 3 Konzept

In diesem Kapitel wird die durchzuführende Aufgabe beschrieben und in Schritte aufgeteilt. Ausgehend von diesen Schritten wird daraufhin das Projekt durchgeführt. Die konkrete Implementierung findet in Kapitel 4 statt.

3.1. Aufgabenstellung

Wie in Kapitel 2 dargelegt, lassen sich über die CloudRail-API Geräte verschiedenen Standards vernetzen. Um Besuchern der Seite den Vorteil der API einfach und zugänglich zu präsentieren, sieht mein Projekt vor, ein Dashboard zu entwerfen. Auf diesem Dashboard werden mehrere Dienste und Geräte parallel direkt steuerbar sein.

Die Aufgabenstellung für das Projekt besitzt folgende Gliederung:

- Zunächst werden geeignete und interessante Geräte/Dienste für die Vernetzung ausgewählt. Hierbei ist es auch wichtig, die genaue Funktion des Gerätes/Dienstes zu bestimmen. (Kapitel 3.3)
- Sind die Geräte und Funktionen gewählt, können diese in der CloudRail-API definiert werden. (Kapitel 4.2)
- Die WunderBar wird eingerichtet und die Sensoren angebracht.(Kapitel 4.3)
- Im Folgeschritt wird ein Grundgerüst des Dashboards entworfen. (Kapitel 4.4)
- Anschließend erfolgt die Integration der API auf das Dashboard. (Kapitel 4.5)
- Zu jeder Kachel werden Code-Ausschnitte der jeweiligen Implementierung zur Verfügung gestellt.
- Abschließend steht das Design des Dashboards.

3.2. Anforderungen und Zielsetzung

Die Anforderung an das Projekt ist, dass das Dashboard zeigt, wie die CloudRail-API das IoE unterstützt. In mehreren Kacheln werden Geräte/Dienste aufgelistet, die unabhängig voneinander agieren sollen. Sprich, ein Post auf Facebook lässt sich schreiben, während der Lautstärkensensor neue Messungen anzeigt. Das Dashboard soll die Vernetzung, sowie Art der Implementierung zeigen. Um Nutzer mehrmals auf die Seite zu locken, werden zusätzlich Funktionen unter den verfügbaren Diensten angeboten. Zum Beispiel kann eine Playlist von Spotify passend zu der Ortstemperatur gesucht werden.

Das Ergebnis wird fester Bestandteil der CloudRail-Homepage werden und zu Marketingzwecken dienen.

Für ein ansprechendes und funktionales Dashboard standen folgende Dashboards als Inspiration.

3.2.1. Sid Lee Dashboard

Sid Lee ist ein französisches Marketing-Unternehmen mit Sitz in Paris. Für eine Kampagne ließen sie sich ein Dashboard entwerfen. Das Ergebnis kann unter `http://dashboard.sidlee.com` betrachtet werden.

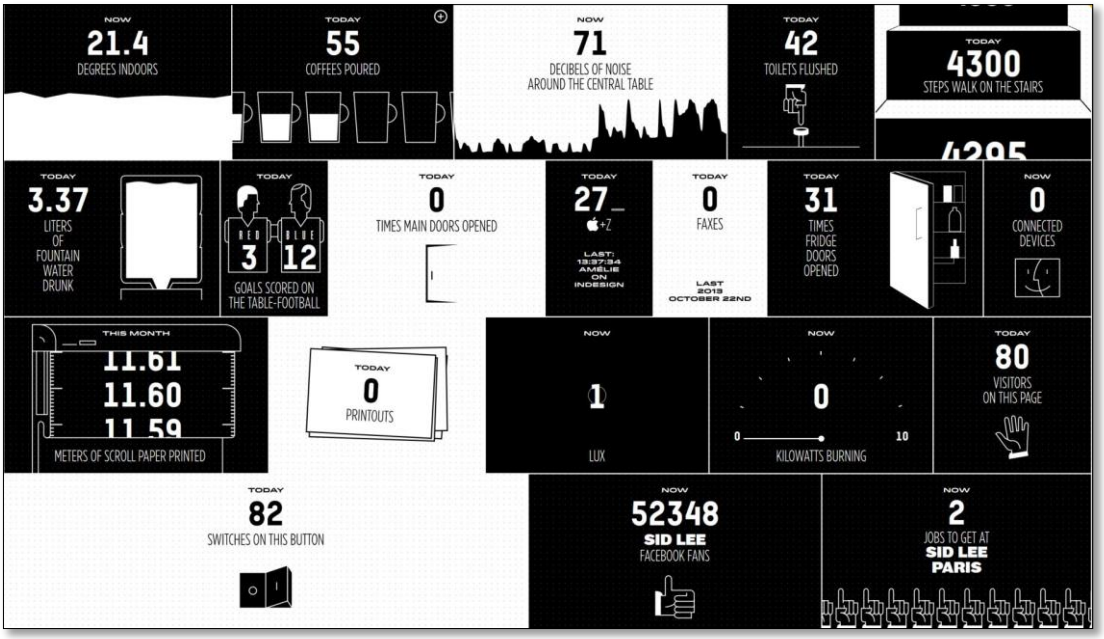


Abbildung 29: Das Dashboard von Sid Lee [63]

Auf dem Dashboard werden einige Daten in Echtzeit aus dem Büro gesammelt und visualisiert. Dazu zählen die Temperatur und Lautstärke, aber auch ausgefallene Ideen, wie die Anzahl an Öffnungen der Kühlschranktür, das aktuelle Ergebnis auf dem Tischkicker oder die Häufigkeit der Nutzung des Undo[28]-Tastenkürzels. Darüber hinaus können zu jeder Kachel Statistiken zum Monat oder der einzelnen Mitarbeiter angezeigt werden. Als Mikrocontroller wurde ein Arduino eingesetzt. Die im Büro verteilten Sensoren senden ihm die gewünschten Daten. Auf dem Arduino können die Daten gefiltert und an das Dashboard weitergeleitet werden.

Ziel des Dashboards ist es, Kunden auf die Seite von Sid Lee aufmerksam zu machen. Das schlichte Design und viele ausgefallene Elemente bringen täglich im Schnitt 200 Besucher auf das Dashboard. Dieses bietet eine gute Grundlage für Marketing-Aktivitäten. Das Konzept, Live-Daten aus dem Büro auf dem Dashboard zu zeigen und zu visualisieren, wird sich deshalb in abgeänderter Art und Weise auch Anwendung auf meinem Dashboard finden.

[28] Das Undo-Tastenkürzel macht die letzte ausgeführte Aktion rückgängig

3.2.2. Vaadin Dashboard

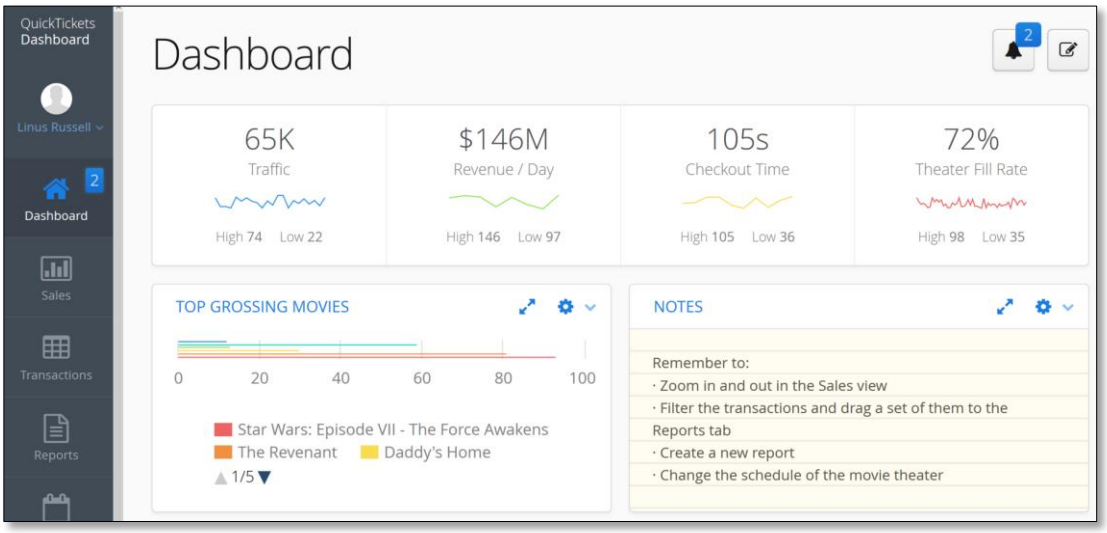


Abbildung 30: Das QuickTickets-Dashboard von Vaadin [64]

Abbildung 30 zeigt das Dashboard von Vaadin. Mit mehr Fokus auf Funktionen und Interaktivität bietet Vaadin ein Beispiel im Zusammenspiel mit dem frei verfügbaren Vaadin Framework. Jede Kachel lässt sich individuell filtern, vergrößern und detaillierter wiedergeben.

Das Dashboard vereinigt Interaktion und visuellen Anspruch. Die dargestellten Daten haben keine Bedeutung, sondern zeigen beispielhafte Auswertungen. Die Bauteile, aus denen das Dashboard erstellt wurde, sind Bestandteil des Frameworks und lassen sich auf diese Weise implementieren.

3.3. Auswahl der Komponenten

Für das Dashboard werden vorwiegend Geräte ausgesucht, die viele Nutzer und einen hohen Bekanntheitsgrad haben. Das Dashboard soll schließlich so vielen Menschen wie möglich von Nutzen sein.

3.3.1. Philips Hue

Die LED-Lampe von Philips kam Oktober 2012 auf den Markt. Sie bietet ein ansprechendes Design und lässt sich über Endgeräte steuern. Zu der Steuerung zählen unter anderem Helligkeit und Lichtfarbe. Es können sogar Szenarios einprogrammiert werden, sprich Farbspiele, die die Hue wiederholend zeigt. Die Philips Hue findet Anwendung im Smart Home Bereich, in der sie die Raumstimmung ändern oder einfach den Lichtschalter ersetzen kann.

Abbildung 31: Komponenten der Philips Hue [65]

Auf Abbildung 31 der vorhergehenden Seite sind die benötigten Komponenten zu sehen. Am linken Bildrand steht die Bridge, die die Verbindung zwischen Lampen, Endgeräten und dem Internet herstellt. So können bestimmte Aktionen ausgeführt werden, die durch Daten aus dem Internet oder dem Endgerät ausgelöst werden. Beispielsweise verfügt das System bereits über ein voreingespeichertes Alarm-Farbspiel, das als Regenalarm eingesetzt werden könnte.

In der Mitte sind die speziellen Hue-Glühbirnen zu sehen. Am rechten Bildrand ist das Endgerät.

3.3.1.1. Steuerung über einen Debugger und Ambieye

Im Folgenden wird dargestellt, wie sich die Philips Hue Lampe über einen Online-Debugger oder das Gratis-Tool Ambieye steuern lässt.

Debugger

Der Debugger lässt sich über die IP der Hue-Bridge aufrufen.

```
http://<IP-Adresse>/debug/clip.html
```

Der Debugger bietet eine leicht verständliche, grafische Oberfläche an. Mit ihm müssen die Befehle händisch eingetragen und Methoden ausgesucht werden. Darum empfiehlt sich der Debugger nur für Nutzer, die mit Request-Methoden und der JSON-Schreibweise vertraut sind.

In das oberste Kästchen wird die URL eingetragen. Der Aufbau des Links ist bei jeder Hue-Lampe und Hue-Bridge gleich:

```
/api/<Nutzer-
ID>/lights/<id>/state
```

Um auf die Lampe zuzugreifen, wird ein Benutzer definiert, im Fall der Abbildung 32 heißt dieser „lightswitch-v4“. Da mehrere Lampen über die Bridge gesteuert werden können, muss im Pfad *lights* die Auswahl der Licht-ID erfolgen.

Der *state*-Pfad bietet nun die Möglichkeit, die Lampe durch API-Calls zu steuern. In den Message-Body werden die Befehle im JSON-Format geschrieben.

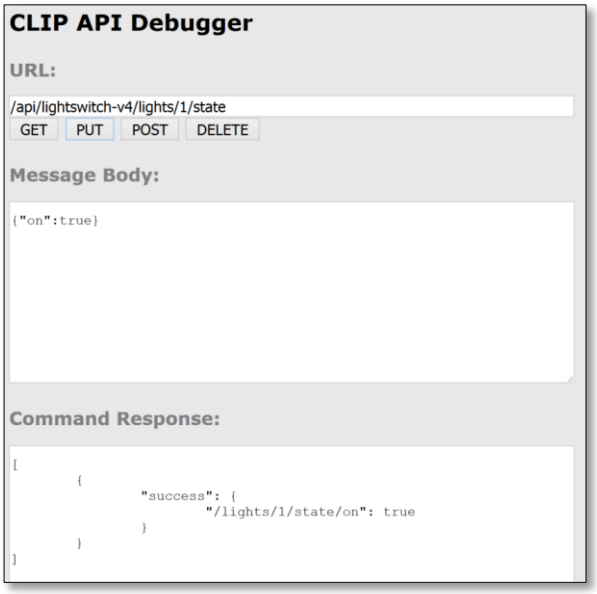


Abbildung 32: Debugger der Hue-Bridge

In der Abbildung 32 wird die Lampe angeschaltet. Dies passiert durch den *on*-Parameter, welcher ein Boolean[29] ist. Bei Angabe von *true* schaltet sich die Lampe ein, mit *false* wird sie ausgeschaltet. Unter der URL sind die zur Auswahl stehenden Methoden. Soll ein Update auf den Lampenzustand erfolgen, ist die PUT-Methode die richtige Wahl.

Als Response liefert die Bridge *error* oder *success* mit einer kurzen Statusmeldung. In der Abbildung 32 ist der Request erfolgreich, die Bridge gibt darüber hinaus den aktuellen Zustand zurück.

Zusammenfassend wird folgende Nachricht per HTTP übertragen:

```
PUT /api/lightswitch-v4/lights/1/state HTTP/1.1
Host: 192.168.8.103

{„on“:true}
```

Durch die PUT-Methode wird der *on*-Parameter aktualisiert. Daneben steht der dementsprechende Pfad. Die IP-Adresse des Host ist die der Bridge. Die eingefügte Adresse steht beispielhaft für eine IP aus dem privaten Netzwerk.

Alternativ zum Debugger können die Befehle auch per Kommandozeile gesendet werden. Folgend wird ein beispielhafter Request mit der PowerShell von Windows durchgeführt.

```
Invoke-WebRequest http://192.168.8.103/api/lightswitch-
v4/lights/ -method GET
```

Mit dem Invoke-WebRequest lassen sich HTTP-Anfragen stellen. Die Antworten können in einer Text-Datei ausgelesen werden. Der oben dargestellte Request ruft den *state*-Pfad der Lampe 1 mit der GET-Methode auf. Diese Methode wird eingesetzt, um Informationen über den Zustand der Lampe aufzurufen.

[29] Wahrheitswert; kann entweder true (wahr) oder false (falsch) sein

Die in der Textdatei ausgelesene Response sieht folgendermaßen aus:

```
{"state":
{"on":false, "bri":254, "hue":31499, "sat":145,
"effect":"none", "xy":[0.2979,0.5002], "alert": "select",
"colormode": "xy", "reachable":true}, "type": "Color light",
"name": "LivingColors 1", "modelid": "LLC010",
"manufacturername": "Philips"}}
```

Die Bridge antwortet mit dem *state*-Pfad. Dieser beinhaltet alle Komponenten zur Steuerung der Lampe, wie *bri* (Brightness – Helligkeit), *hue* (Hue – Lichtfarbe), *sat* (Saturation – Sättigung) und *colormode* (Colormode – Farbmodus). Mit letzterem können Farbspiele realisiert werden. Die Response zeigt, welche Werte der Nutzer anpassen kann. Wünscht er beispielsweise eine höhere Helligkeit, kann er den Wert von *bri* mithilfe des PUT-Request erhöhen.

Ambieye

Für Anwender ohne technische Vorkenntnisse gibt es eine interessante Alternative. Ambieye bietet eine grafische Oberfläche, wodurch sich die Lampe mit Buttons und Reglern steuern lässt.

Ambieye funktioniert ähnlich wie der Debugger. Das Tool lässt sich durch eine URL aufrufen und mit der Bridge verbinden. Hierzu ist es lediglich notwendig, sich mit der Bridge und dem Endgerät im gleichen Netzwerk zu befinden. Auch Ambieye schickt Requests und Responses. Nutzer können diese in der Konsole auslesen.

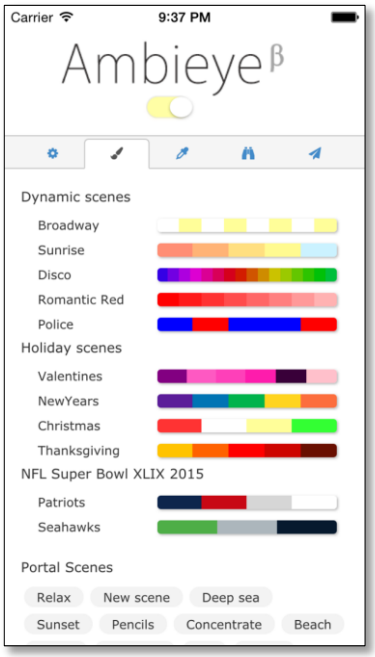


Abbildung 33: Die grafische Oberfläche von Ambieye [66]

In mehreren Unterkategorien bietet Ambieye eine leicht verständliche Benutzeroberfläche. In *Scenes* lassen sich Farbspiele realisieren und in *Colors* (siehe Abb. 33) können konkrete Farben ausgewählt werden. Unter dem Titel kann die Lampe direkt ein- oder ausgeschalten werden. Durch die einfache Bedienung und die große Auswahl an Funktionen ist Ambieye die bessere Alternative zum Debugger.

3.3.2. Nest Thermostat

Abbildung 34: Nest Thermostat [67]

Das Raumthermostat von Nest, mittlerweile aufgekauft von Google, passt Raumtemperaturen automatisch an. Es wurde bereits mehrfach in früheren Kapiteln dieser Arbeit als Musterbeispiel für automatisierte Objekte erläutert.

Um die Erläuterungen kurz aufzugreifen, konfiguriert das Nest Thermostat die Zimmertemperaturen eigenständig. Dies wird dadurch realisiert, dass es die Einstellungsmuster der Nutzer innerhalb einer Woche lernt und anwendet. Falls niemand Zuhause ist, fährt es eigenständig herunter und spart so Energie. Verschiedene Sensoren sorgen für die Daten. Im Thermostat sind Temperatur- und Feuchtigkeitssensor, Nahfeld-, sowie Fernfeldaktivitätensensor eingebaut. Sobald sich der Nutzer dem Thermostat nähert, kann das Gerät ihn erkennen und das Display einschalten. Der Zugriff durch Endgeräte erfolgt über WiFi. [68] Mit dem Smartphone oder Computer lassen sich per App Temperaturen einsehen, einstellen, sowie aktive Thermostate anzeigen.

Im Gegensatz zur Philips Hue, müssen Nutzer des Nest Thermostates nicht im gleichen Netz sein, um das Gerät zu kontrollieren. Der Zugriff erfolgt online oder in einer App über einen Nutzer-Account.

Aufbau der API-Calls

Für den Zugriff auf das Thermostat ist also eine Authentifizierung notwendig. Zunächst muss für die Anwendung eine ID und Passwort generiert werden. Dies passiert durch die Anmeldung der Anwendung auf der Nest-Plattform (siehe Abb. 35).

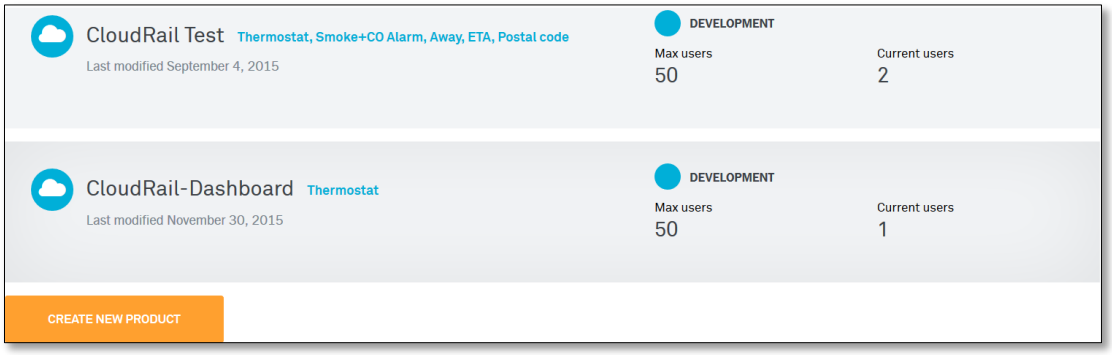


Abbildung 35: Nest Produkte [69]

Angelegt sind ein Test-Produkt und das Dashboard. In diesen Produktbeschreibungen müssen die Berechtigungen der Anwendung, Kategorie, Anzahl der Nutzer, eine Support-URL und die Redirect-URI festgelegt werden. Im Gegenzug generiert Nest die ID und das Passwort. Diese Daten werden beim Authentifizierungsprozess abgefragt und müssen deshalb einmalig übergeben werden.

Funktionen der API sind die Ausgabe aller vorhandenen Geräte (*Get thermostats*), die Ausgabe der Temperatur eines bestimmten Gerätes (*Get target temperature*) und die Eingabe der Temperatur eines bestimmten Gerätes (*Set target temperature*). Für die beiden letzteren, wird eine Thermostat-ID benötigt, welche durch die erste Methode angefragt werden kann. Für die Anfragen der Temperatur-Ein- oder Ausgabe muss also vorher eine Thermostat-Abfrage durchgeführt werden. Nach dem Authentifizierungsprozess, werden die Thermostate durch folgenden Request aufgerufen:

```
GET /structures?auth=<access-token> HTTP/1.1
Host: https://developer-api.nest.com
```

Mit jedem Request muss, der Authentifizierung bedingt, der Access-Token mitgeschickt werden. In diesem Fall wird er als *auth*-Query an die URL angehängt. In der Response sendet Nest die vorhandenen Thermostate mit IDs. Diese IDs können nun für weitere Anfragen eingesetzt werden.

Get target temperature:

```
GET /devices/thermostats/<thermostat-id>?auth=<access-
token>HTTP/1.1
Host: https://developer-api.nest.com
```

Set target temperature:

```
PUT /devices/thermostats/<thermostat-id>?auth=<access-
token>HTTP/1.1
Host: https://developer-api.nest.com/

target_temperature_c=17.5
```

Die Temperatur wird im Body des Request übergeben. Die Einheit kann als Celsius (`target_temperature_c`) oder Fahrenheit (`target_temperature_f`) festgelegt werden.

3.3.3. Netatmo

Abbildung 36: Netatmo Wetterstation [70]

Das Netatmo hebt sich im Markt der Wetterstationen von seinen Konkurrenten durch eine Vielzahl an Funktionen ab. Es besteht aus zwei Modulen – Indoor und Outdoor. Das Indoor-Modul wertet das Klima im Haus aus. Dazu zählen Temperatur, Feuchtigkeit, Luftqualität, CO_2 und Lautstärke. Mittels Internet-Anbindung und Smartphone kann Netatmo die gemessenen Werte analysieren und visualisieren. So gibt es eine Warnmeldung bei zu hohen CO_2-Werten und Hinweise zum Luftqualitätsindex. Mit dem Outdoor-Modul können ebenfalls Temperatur, Feuchtigkeit und Luftqualität, aber auch Luftdruck und Wetterdaten gesammelt werden. Die Grundausrüstung um ein Indoor- und Outdoor-Modul kann um drei weitere Outdoor-Module, einen Regenmesser und einen Windmesser erweitert werden.

Die Verbindung zwischen Endgerät und Station entsteht via WiFi. Auf dem Netatmo-Server können außerdem zu jeder Zeit die momentanen Werte eingesehen oder Messwertkurven mit vergangenen Werten aufgerufen werden. [71]
Zugriff und Steuerung erfolgen wie beim Nest Thermostat online oder per App durch einen Nutzeraccount.

Aufbau der API-Calls
Die Netatmo-API verlangt beim Authentifizierungsprozess ebenso wie die Nest-API eine Anwendungs-ID, ein Passwort und eine URI, auf die sie zurückleiten soll. Diese Daten werden wieder durch die Anmeldung der Anwendung auf der Netatmo-Seite generiert. Es können mehrere Anwendungen gleichzeitig angelegt werden. Neben Angabe der Website, Redirect-URI, eines Namen und Beschreibung kann auch ein Bild eingefügt werden, das der Nutzer zusammen mit der Beschreibung bei der Authentifizierung zu sehen bekommt.

Netatmo kommt mit zwei Methoden: der Abfrage einer Geräteliste (*Get Devicelist*) und der Abfrage der Klimadaten eines bestimmten Gerätes (*Get Measure*). Für die Abfrage der Klimadaten ist die ID eines Gerätes Voraussetzung. Darum ist auch hier der Aufruf der Methode *Get Devicelist* vor der Methode *Get Measure* ein Muss.

Get Devicelist:

```
GET /devicelist?access_token=<access-token> HTTP/1.1
Host: https://api.netatmo.net/api
```

Die Response liefert gerätebezogene Daten und die IDs. Damit können nun die Klimadaten abgerufen werden.

Get Measure:

```
GET  /api/getmeasure?access_token=<access-
token>&device_id=<netatmo-
id>&type=Temperature,Humidity,Co2&optimize=false&scale=30min
HTTP/1.1
Host: https://api.netatmo.net
```

Im Request werden Access-Token, die Geräte-ID, der Typ der Response, eine Response-Optimierung für Mobile-Anwendungen und eine Skalierung definiert. Der Typ der Response kann, je nach Ausstattung der Netamo-Station, folgende weitere Werte annehmen: `Noise`, `Rain`, `WindStrength`, `WindAngle`, `GustStrength`, `GustAngle`. Der Boolean *optimize* gibt an, ob eine Optimierung der Response auf mobile Endgeräte vorgenommen werden soll. Im Fall meines webbasierten Dashboards muss dieser Wert also auf *false* gesetzt werden. Mit *scale* kann das Intervall der Messungen festgelegt werden.

Als Response liefert Netatmo die im Typ festgelegten Daten als JSON-Objekt.

3.3.4. WunderBar

Die WunderBar von relayr bietet eine Plattform in Schokoladen-Tafel Optik. Die Stücke lassen sich abbrechen und an verschiedensten Orten platzieren. Die Sensoren der WunderBar liefern Auswertungen in Echtzeit. Die Kommunikation zwischen Sensoren und Hauptplatine erfolgt über das in Kapitel 1.2.2 vorgestellte Bluetooth Smart. Die Platine selbst wird über WLAN angesteuert. Die Auswertungen lassen sich durch eine verfügbare App abrufen oder durch herunterladbare SDKs in die eigene Anwendung einbauen. Durch eine Batterie ist keine Stromversorgung nötig, sodass die Sensoren in jeder Ecke angebracht werden können. Zu den Sensoren zählen [73]:

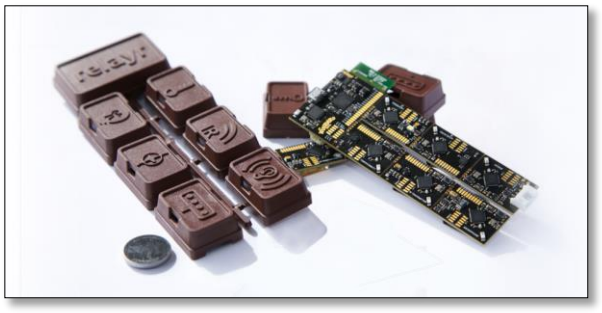


Abbildung 37: Die WunderBar von relayr [72]

- **ein Temperatur- und Feuchtigkeitsmesser**
  Damit können relative Luftfeuchtigkeit und Temperaturen von -40° bis 125° C gemessen werden. Relayr verspricht eine maximale Abweichung von +/- 0,3°C.
- **einen Lichtfarben- und Näherungssensor**
  Mit dem eingebauten TAOS TCS3771 von AMS Technologies kann Licht in seine RGB-Farben aufgelöst werden. Außerdem verfügt das Modul über einen Näherungssensor (PROX), mit dem sich nähernde Objekte identifiziert werden können, ohne einen physischen Kontakt zu benötigen.
- **einen Beschleunigungssensor und Gyroskop**
  Das Gyroskop ist ein Lagesensor, das die Bewegung durch 6 Achsen bestimmt.
- **einen Infrarottransmitter**
  Mit entsprechendem IR-Responder können Objekte angesteuert werden. Beispielsweise dient der Transmitter als Fernbedienung für den TV oder das Garagentor.
- **ein Mikrophon**
  Das Mikrophon kann Geräuschlevel bis zu 10 Bit darstellen. Die Ausgabe erfolgt nicht in Dezibel, sondern in festgelegten Geräuschstufen.
- **eine Bridge**
  Durch die Bridge können weitere Sensoren angeschlossen werden. Die Verbindung zu einem Arduino oder Raspberry ist ebenso denkbar.

Onboarding

Für die erste Nutzung der WunderBar müssen die Sensoren initialisiert und identifiziert werden. Zum Zeitpunkt der Erstellung der Thesis war dies lediglich durch eine Android-App möglich. Folgende Schritte müssen für das Onboarding durchgeführt werden:

- **Anbringen der Batterie und Verbinden des USB-Kabels**
  An das Hauptmodul der WunderBar muss zunächst eine Batterie angebracht werden. Da diese anfangs nicht komplett geladen ist, sollte eine zusätzliche Stromversorgung per USB vorhanden sein.
- **Download und Aufruf der relayr App**
  Die relayr App gibt es im Google Play Store für Android und bei iTunes für iOS-Geräte zum kostenlosen Download
- **Konfigurieren der WunderBar**
  Ist die App gestartet, kann direkt eine eigene WunderBar angelegt werden. Dieser muss nur ein Name gegeben und mit demselben Netz des zugreifenden Endgerätes verbunden werden.
- **Initialisierungsprozess**
  Das eigentliche Onboarding findet jetzt statt. Jeder Sensor des Moduls verbindet sich über Bluetooth Smart mit dem Hauptmodul und wird identifiziert.
- **Auslesen der Daten**
  Ist alles gut gelaufen, verbindet sich das Hauptmodul durch das WLAN mit der Cloud und zeigt erste Messungen in der App an. Im Browser können die Daten durch ein Dashboard aus der Cloud abgerufen werden (siehe Abb. 38).

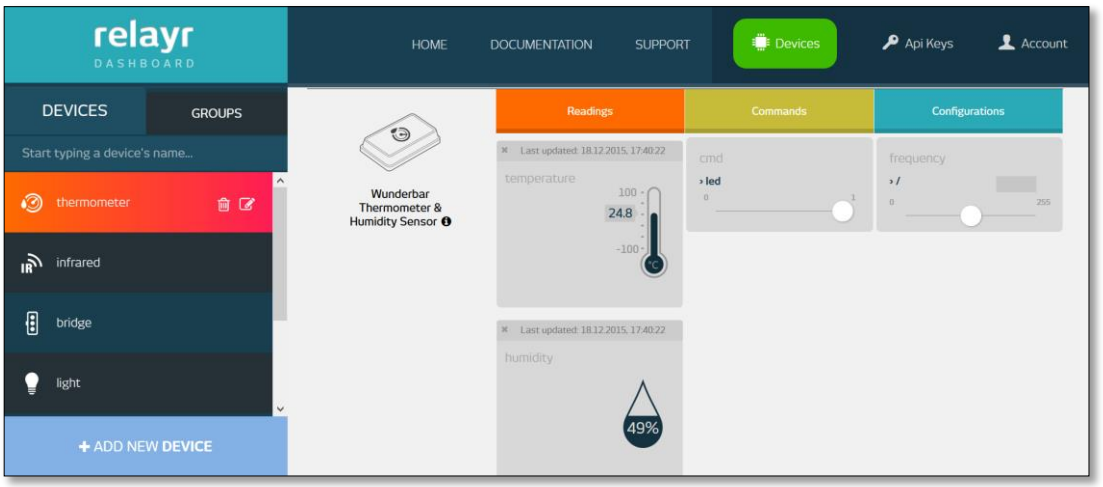


Abbildung 38: Das Dashboard von relayr [74]

3.3.5. OpenWeather

OpenWeather ist ein Online-Wetterdienst. Mit der API können Nutzer aktuelle Wetterdaten und Wettervorhersagen abrufen. Die API benötigt dabei keine Authentifizierung. Anfragen können per HTTP Request an die URL `http://api.openweather.com` erfolgen.

Aufbau der API-Calls
Die Requests beinhalten entweder den Stadtnamen oder die PLZ mit Landescode. Beide Anfragetypen müssen einen API-Key (App-ID) mitsenden. Dieser wird durch eine Anmeldung bei OpenWeather generiert. Durch den Key kann OpenWeather die Zahl an Calls nachverfolgen und ggf. dem Nutzer Kosten anrechnen.

Ein beispielhafter Request wird wie folgt durchgeführt:

```
GET     /data/2.5/weather?zip=77652,DE&APPID=93ba8bb89a0beb6492
HTTP/1.1
Host: http://api.openweathermap.org
```

Die Methode wird wie bei allen Abfragen auf GET gesetzt. Der Pfad bezieht sich auf die Datenbank (*data*) von OpenWeather mit der aktuellen Versionsnummer der API (*2.5*). An *weather* werden nun die Queries angehängt. Die Anfrage bezieht sich auf die Stadt mit der Postleitzahl 77652 aus Deutschland, also Offenburg.

Die Response ist wie folgt aufgebaut:

```
{"coord":{"lon":7.93,"lat":48.48},"weather":[{"id":701,"main":
"Mist","description":"mist","icon":"50d"}],"main":{"temp":281.
15,"pressure":1033,"humidity":87,"temp_min":281.15,"temp_max":
281.15},"wind":{"speed":0.5},"clouds":{"all":40},"country":"DE
","sunrise":1449299021,"sunset":1449329637},"id":2857798,"name
":"Offenburg","cod":200}
```

Die Response beinhaltet detaillierte Daten über die angefragte Stadt, wie Längen- und Breitengrad, aktuelle Wetterlage (Temperatur, Luftdruck, Luftfeuchtigkeit, Wind, Bewölkung) sowie Zeitpunkt des Sonnenauf- und Sonnenunterganges.

3.3.6. Facebook

Facebook wird mit der Post-Funktion Anwendung auf dem Dashboard finden. Der Nutzer kann damit einige abgerufene Daten direkt auf seine oder die Dashboard-Pinnwand weiterleiten. Der Aufbau und die Art und Weise der Requests und Responses wurde bereits in Kapitel 1.8.10 ausführlich erläutert.

3.3.7. Spotify

Die Spotify API stellt mehrere Methoden zur Verfügung, mit denen der Musikkatalog durchsucht, Playlists erstellt oder Songs angehört werden können. Spotify benötigt einen Authentifizierungsprozess. Die Anwendungs-ID und ein Passwort können durch das Anmelden der Anwendung auf Spotify generiert werden.

Aufbau der API-Calls
Das Suchen von Künstlern und Liedern wird durch den Pfad *search* realisiert. Angepasst werden muss jeweils nur das Query *type*. In *q* wird das Suchkriterium eingetragen.

```
GET /v1/search?type=track&q=Let+it+be HTTP/1.1
Host: https://api.spotify.com
```

```
GET /v1/search?type=artist&q=Beatles HTTP/1.1
Host: https://api.spotify.com
```

Als Response übermittelt der Server sämtliche Daten in einem Array. In diesem sind mehrere Antworten auf das Suchkriterium vorhanden, welche in Künstler, Album und Song aufgeteilt sind.

Um den eingeloggten Nutzer abzurufen, wird folgender Request verwendet.

```
GET /v1/me HTTP/1.1
Host: https://api.spotify.com
```

Das Abrufen einer bestimmten Playlist eines bestimmten Nutzers wird durch die Angabe der jeweiligen IDs realisiert. Die IDs können in Spotify nachgeschlagen oder per API-Request angefragt werden.

```
GET /v1/users/<user-id>/playlists/<playlist_id> HTTP/1.1
Host: https://api.spotify.com
```

3.4. Festlegen der Kachel-Funktionen

Im Folgenden wird den ausgewählten Komponenten ihre Funktion auf dem Dashboard zugeordnet. In dieser Reihenfolge wird in Kapitel 4.5 die Implementierung beschrieben.

1. **Facebook**
   Der Nutzer kann in einem Textfeld auf seine eigene Pinnwand posten oder einen Kommentar an die CloudRail-Dashboard-Pinnwand verfassen
2. **Philips Hue**
   Die Steuerung der Helligkeit, Auswählen der Farbe, sowie das An- und Ausschalten der Hue-Lampe sind Teil der Hue-Kachel
3. **Nest Thermostat**
   Der Nutzer kann hier die Temperatur seines Thermostats einsehen und neu einstellen
4. **Netatmo**
   Mit einem einzigen Button kann der Nutzer seine Klimawerte abfragen. Diese kann er direkt auf Facebook teilen
5. **OpenWeather**
   Durch die Eingabe des Stadtnamens oder der Postleitzahl kann der Nutzer aktuelle Wetterdaten abrufen und direkt auf Facebook posten
6. **Spotify**
   Die Suche nach Artisten oder Songs und das Anhören von eigenen oder vorgeschlagenen Playlists zählen zu den Funktionen der Kachel
7. **Ortstemperatur auf der Hue visualisieren**
   Der Nutzer kann durch die Eingabe seines Heimatortes die Temperatur auf der Hue-Lampe darstellen. Warme Temperaturen führen zu kühleren und bläulichen Farben, während kalte Temperaturen zu rötlicheren und warmen Farben führen.
8. **Playlist passend zum Wetter**
   Die Ortstemperatur der vom Nutzer eingegebenen Stadt führt zu einer passenden Playlist
9. **Kaffezähler**
   Die Höhe des Kaffeekonsums im Team kann eingesehen werden
10. **Thermostat**
    Auf dieser Kachel wird die Temperatur aus dem Büro dargestellt
11. **Lautstärkemesser**
    Die Lautstärke aus dem CloudRail-Büro in Mannheim kann in Echtzeit betrachtet werden
12. **Workbench-Verlinkung**
    Auf der letzten Kachel wird dem Besucher ein Direkt-Link zur Workbench angeboten mit der Aufforderung, nun selbst aktiv zu werden.

3.5. Erste Skizze

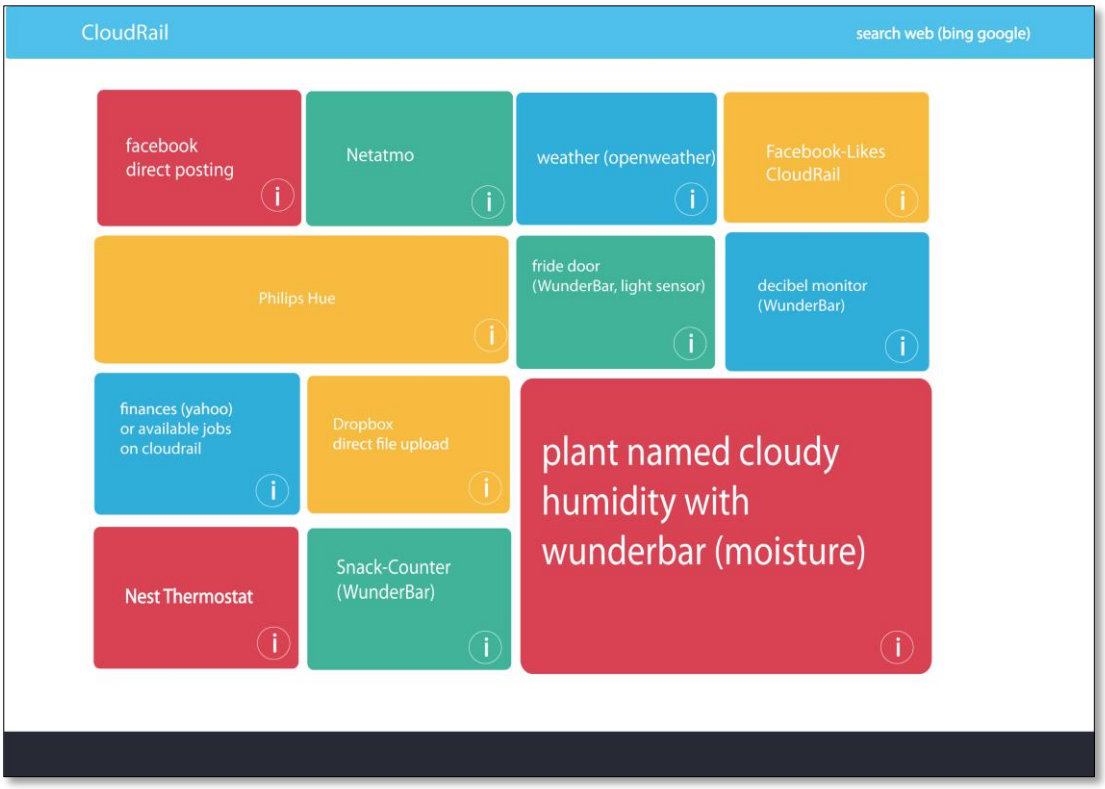


Abbildung 39: Skizze für den Aufbau des Dashboards

Die Abbildung 39 zeigt eine erste Idee für das Dashboard. Jede Kachel besitzt einen Dienst und die damit verbundenen Funktionen.
Wie in den Anforderungen festgelegt sind sowohl Web-Services wie Facebook und Dropbox, Geräte wie die Philips Hue, als auch Sensoren der WunderBar für Live-Auswertungen vertreten. Zu jeder Kachel sollen Informationen ausgegeben werden können. Diese können über den i-Button als Pop-Up angezeigt werden.

Kapitel 4 Implementierung

Für die Implementierung werden die Schritte, wie im Konzept angegeben, abgearbeitet. Die Geräte wurden bereits ausgewählt. Die individuellen API-Calls wurden beschrieben. Nun können diese in der CloudRail-Workbench definiert werden.

4.1. Integration von Facebook

Um die Facebook-API in die CloudRail-Workbench zu integrieren, wird eine API-Dokumentation benötigt. In API-Dokumentationen stehen jegliche Informationen, die der Entwickler für die Nutzung der API benötigt, wie bspw. die Basis-URL, Requests und Responses, sowie Methoden. Unter `https://developers.facebook.com/` können eingeloggte Nutzer diese Informationen abrufen.

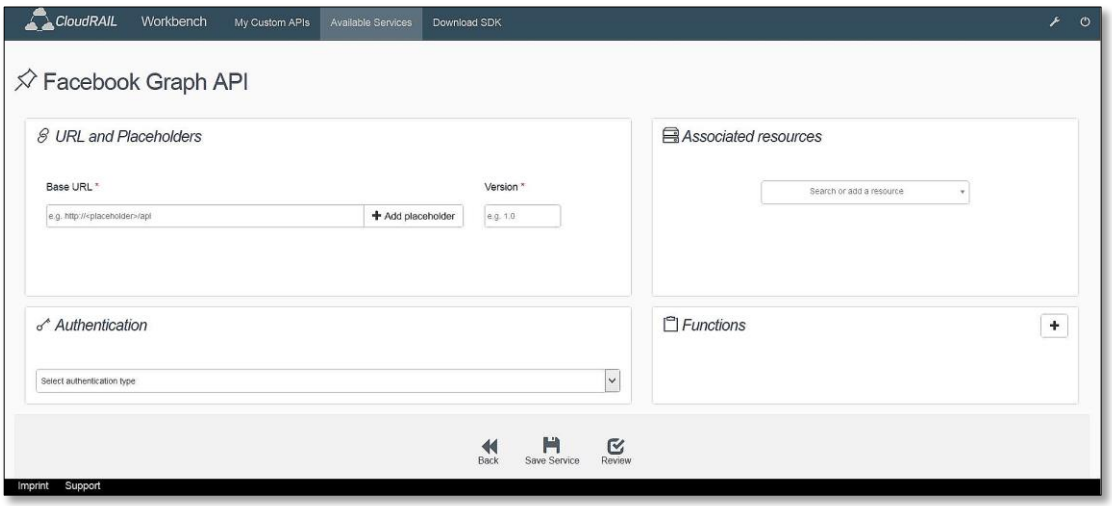


Abbildung 40: API-Definition auf der Workbench[17]

Zunächst wird auf der Workbench ein Name für die API ausgewählt. In diesem Fall wurde er auf Facebook Graph API festgelegt. Die Abbildung 40 zeigt das Fenster, in dem die Definition durchgeführt wird.

Für die Erstellung der API müssen vier Felder bearbeitet werden. Zunächst werden die Basis-URL und globale Platzhalter definiert (*URL and Placeholders*). Dann folgt die Authentifizierung. Facebook authentifiziert Nutzer über das OAuth-Verfahren. (*Authentication*). Anschließend wird die Ressource in das System eingetragen. Dabei wird eine Kategorie festgelegt, um später Nutzern das Suchen und Finden von APIs zu erleichtern, sowie mehrere Dienste mit der API zu assoziieren. (Associated resources). Zuletzt können Methoden definiert werden, die durch die API ausgeführt werden (*Functions*).

Folgend wird Schritt für Schritt die Bearbeitung dieser Felder erläutert. Als Beispiel wird eine Methode implementiert, die alle Pinnwandeinträge des Nutzers ausgeben soll.

4.1.1. URL and Platzhalter

Die Basis-URL ist in der Dokumentation des Herstellers vorgegeben. Auch die Versionsnummer, die im Laufe der Entwicklung geändert werden kann, ist der Dokumentation zu entnehmen.

Globale Platzhalter ergeben sich aus der Authentifizierung oder durch veränderbare Aufrufe, wie es bspw. bei Philips Hue durch die individuelle IP-Adresse der Fall ist. Die Abbildung 41 zeigt die eingetragene URL, Versionsnummer und alle globalen Platzhalter. Dazu zählen Identität des Client (*app-id*), die URI zu der zurückgeleitet werden soll (*redirect-uri*), die Berechtigungen, die der User akzeptieren muss (*scope*), das Passwort des Clients (*app-secret*) und ein CSRF-Token (*state*). Das Token schützt die Calls vor CSRF (Cross-Site Request Forgery)-Attacken. CSRF-Attacken betreffen authentifizierte Nutzer und versuchen den Login-Status auszunutzen. Im Fall von Facebook könnte der Angreifer Passwörter ändern und im Namen des Nutzers Aktionen ausführen.

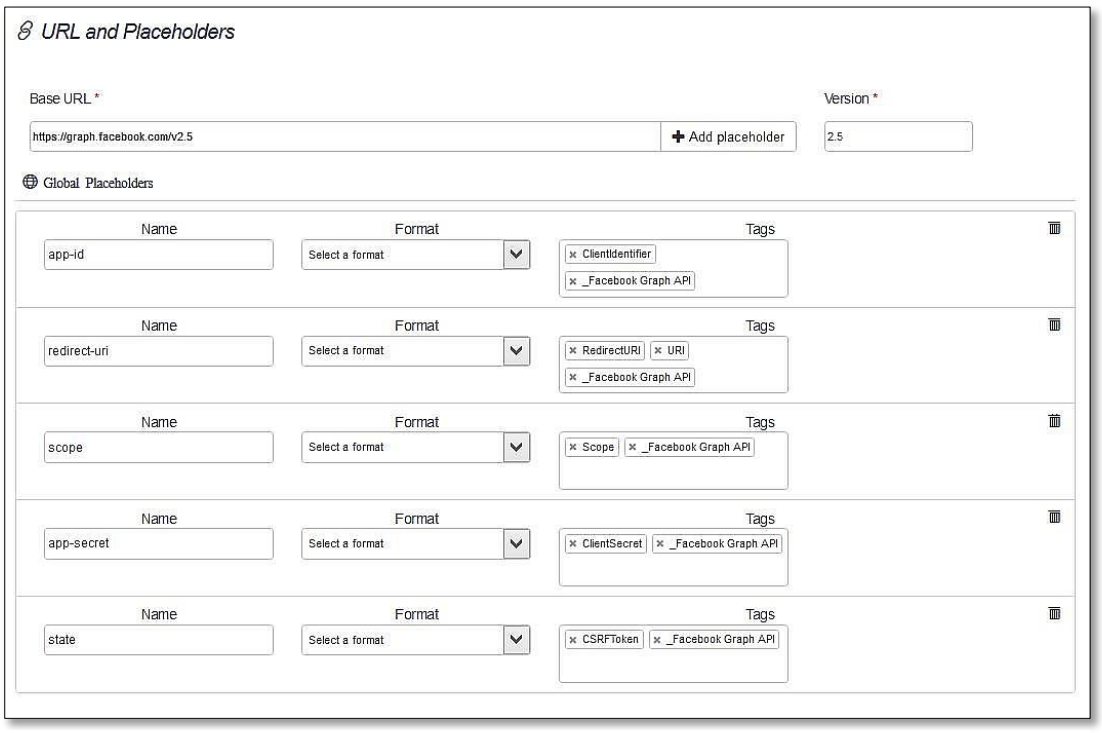


Abbildung 41: Definition der URL und Platzhalter [17]

Die globalen Platzhalter haben allesamt ihren Ursprung in der Authentifizierung.

4.1.2. Authentifizierung

Wie erwähnt gibt Facebook das OAuth 2.0, also eine neue Version des OAuth-Verfahrens als Authentifizierung vor. In der Dokumentation lassen sich die URLs für die Authentifizierung auslesen und eintragen (siehe Abb. 42). Durch die Authorization-URL loggt sich der User auf dem Client ein. Der Client schickt seine ID, eine Redirect-

URI, die durch den User zugelassenen App-Berechtigungen, den Response-Type[30] und das CSRF-Token an Facebook.

Wie im Code-Verfahren (Kapitel 1.8.9) erklärt, generiert Facebook einen Code, der vom Client in einen Access Token umgetauscht werden kann. So können Client und User eindeutig identifiziert werden.

Da die Access Tokens nur eine beschränkte Lebensdauer haben, können sie nach einer Weile erneuert werden. Dies passiert durch die Refresh Token URL, durch die der Client erneut seine ID und sein Passwort an Facebook sendet und einen neuen Access Token anfordert.

Abbildung 42: Authentifizierungs-URLs [17]

4.1.3 Zugehörige Ressource

In diesem Feld wird der Dienst als solcher registriert. Sollten mehrere Dienste auf die API zugreifen wollen, können sie in diesem Feld hinzugefügt werden. Durch das Eintragen einer Kategorie und des Herstellernamen können die Dienste eindeutig getrennt werden. Abbildung 43 zeigt beide ausgefüllten Felder.

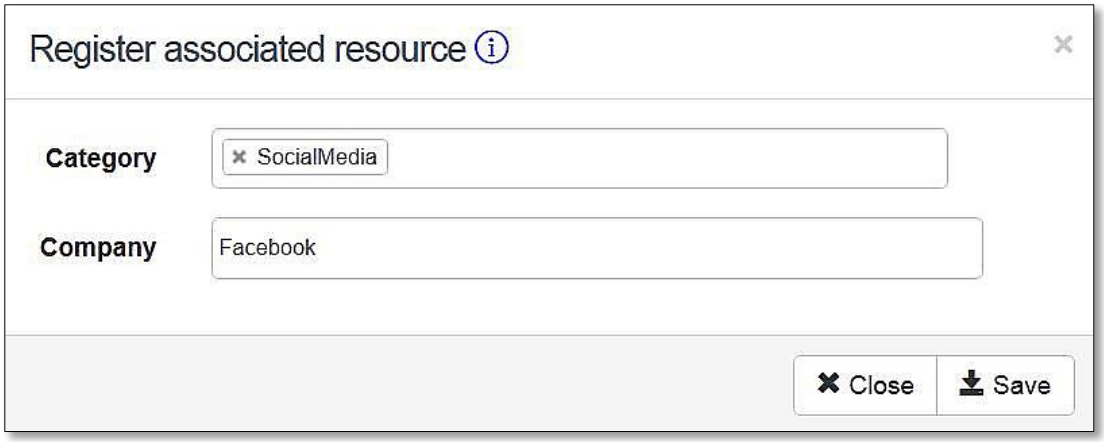


Abbildung 43: Registrierung zugehöriger Ressourcen [17]

[30] Facebook unterstützt Code oder Token als Response Type. In CloudRail ist momentan nur der Code-Flow zugelassen.

4.1.3. Funktionen

Im Funktionen-Feld können neue Methoden angelegt werden. Diese Methoden werden später durch die API ausgeführt. Im Beispiel wird der Feed des Users abgerufen.

4.1.3.1. Pfad und Platzhalter

Zunächst wird hierzu der Pfad angegeben, der nach der Basis URL aufgerufen wird. Weiterhin können Queries und lokale Platzhalter eingetragen werden. Um den Feed abzurufen, sind keine Queries nötig. Ein lokaler Platzhalter ergibt sich aus der Notwendigkeit, den Access Token in jedem Request-Header mitzuschicken. Abbildung 44 zeigt den Pfad und die Platzhalter.

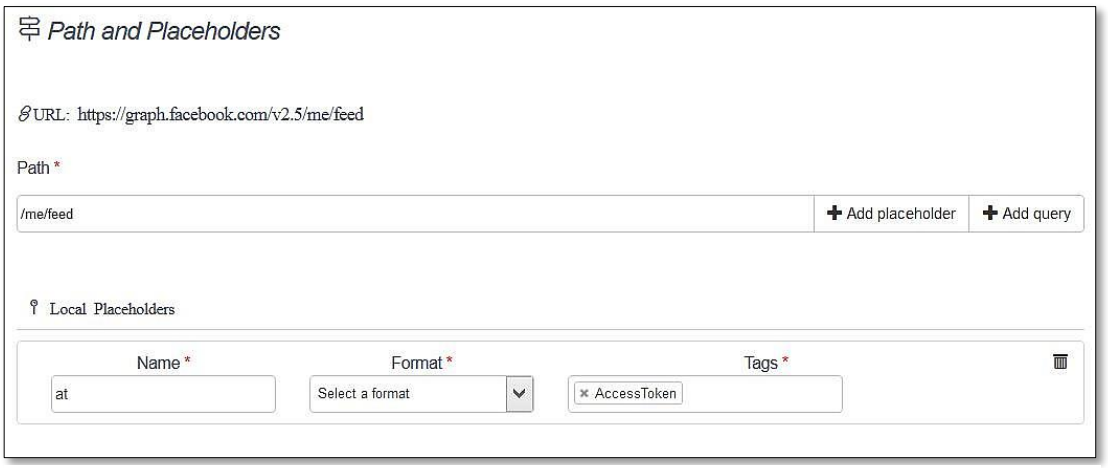


Abbildung 44: Angabe des Pfades und der Platzhalter [17]

4.1.3.2. Header

Für den Header können zusätzliche Informationen wie der Access Token oder Location angegeben werden. Facebook erwartet mit jedem Request den Access Token, um Nutzer und angegebene Berechtigungen auslesen zu können. Abbildung 45 stellt das ausgefüllte Headers-Feld dar, in dem der Token als Platzhalter (<*at*>) definiert wird. Dieser Platzhalter wird im späteren Autorisierungsprozess durch den individuellen Access Token ersetzt.

Abbildung 45: Einfügen von Request-Headern [17]

4.1.3.3. Eigenschaften

Unter Properties werden die HTTP-Methode des Request, Art und Aufbau des Request-Body, sowie des Response-Body eingetragen. Als Body-Type wird momentan nur JSON angeboten, XML soll folgen. Die HTTP-Methode ist in diesem Beispiel GET, da Daten aus Facebook abgerufen werden sollen. Der Request-Body ist leer, da keine Parameter an Facebook übertragen werden müssen. Im Response-Body muss nun der Aufbau der JSON-Form angegeben werden.

Eine beispielhafte Response zur Anfrage des User-Feed sieht so aus:

```
{
 "data": [
      {
       "message": "Glückwunsch Max ;)",
       "created_time": "2014-09-03T18:52:44+0000",
       "id": "849041278451201_825564410798888"
     },
       {
       "story": "Max Mustermann updated his profile picture.",
       "created_time": "2013-11-22T21:20:51+0000",
       "id": "849041278451201_674407922581205"
     }
 ]
}
```

Aus der Response lässt sich der Aufbau jeder weiteren Response erkennen. Ein solches Muster muss nun in die Workbench integriert werden.

Die Response wird in ein JSON-Objekt verpackt. Dies zeigen die äußeren geschwungenen Klammern. Das Objekt beinhaltet das Array *data*. Ein Array ist hier sinnvoll, da die Anzahl an Einträgen je nach Nutzer variieren. Durch einen Array können beliebig viele Einträge hinzugefügt werden.

Im Array sind also die Einträge des User Feed erneut als JSON-Objekt vorhanden. Die erste Variable *message* steht für einen schriftlichen Eintrag auf der Pinnwand. Im Objekt sind des Weiteren das genaue Erstelldatum, also der Zeitpunkt des Eintrages, und eine von Facebook erstellte User- und Post-ID. Die Variable *message* kann durch *story* ersetzt werden, wenn es sich bei dem Eintrag um eine Aktivität bezüglich Fotos, Events oder Verlinkung handelt.

Der Aufbau der JSON-Response muss also wie folgt angegeben werden:

```
{     "format": "JSON",
      "type": "Object",
      "required": ["data"],
      "properties": {
            "data": {
                  "type": "Array",
                  "tags": ["Data"],
                  "items": [{
                        "type": "Object",
                        "required": ["created_time", "id"],
                        "properties": {
```

```
"story": {
      "type": "String",
      "tags": ["Story"]
      },
"message": {
      "type": "String",
      "tags": ["Message"]
},
"created_time": {
      "type: "String",
      "tags": ["CreatedTime"],
},
"id": {
      "type": "String",
      "tags": ["Identifier"]}}}]}}}
```

Mit diesem Aufbau kann CloudRail nun jede Response richtig darstellen.

Hat man alle Funktionen definiert, werden diese im Feld Functions aufgeführt. Abbildung 46 zeigt verschiedene Methoden, die nun implementiert wurden. *Get User Feed* wurde im Beispiel erläutert. Weitere Methoden sind das Teilen von Inhalten an die eigene Pinnwand oder an eine Seite und das Abrufen der vom User verwalteten Seiten.

Abbildung 46: Funktionen von Facebook [17]

4.2. Vollständige Dokumentation der weiteren APIs

In den folgenden beiden Tabellen wird aufgeführt, wie die APIs in CloudRail definiert werden.

Hierzu die Legende:

- **App-ID:**
  Steht für die Client-ID, die bei der Autorisierung der Anwendung bei dem jeweiligen Hersteller für diese generiert wird
- **App-Secret:**
  Steht für das zugehörige, ebenfalls generierte Passwort
- **Redirect-URI:**
  Beinhaltet die URI, auf die die API zurückleitet; ist also die URI der Dashboard-Anwendung
- **State:**
  Beinhaltet das CSRF-Token
- **Scope:**
  Gibt die Berechtigungen an, die der Nutzer für die Nutzung der Anwendung gewährleisten muss

Service	*Basis-URL*	*Authentifizierung*	*Platzhalter (global)*	*Funktionen*	*Dokumentation*
Facebook	https://graph.facebook.com/v2.5	+	App-ID; Redirect-URI; Scope; App-Secret; State	Post To User Feed; Post As Page	https://developers.facebook.com
Nest	https://developer-api.nest.com	+	App-ID; App-Secret; State	Get thermostats; Get target temperature; Set target temperature	https://developer.nest.com/
Netatmo	https://api.netatmo.net/api	+	App-ID; Redirect-URI; State; App-Secret	Get Devicelist; Get Measure	https://dev.netatmo.com/doc
OpenWeather	http://api.openweathermap.org/data/2.5/weather	-	-	Get Weather By Zipcode; Get Weather By Cityname	http://openweathermap.org/api
Philips Hue	http://<ip_address>/api	-	IP-Adresse	Set Light State	http://www.developers.meethue.com/
Spotify	https://api.spotify.com/v1/	+	App-ID; Redirect-URI; State; Scope	Search for Artist; Search for Track; Browse featured playlists; Browse own playlist	https://developer.spotify.com/

Tabelle 1: Integration der Services

Methode	*URL*	*Queries*	*Platzhalter*	*Header*	*Art*	*Request*	*Response*
Post To User Feed	/me/feed	-	Access Token	Authorization	POST	Message	Post-ID
Post As Page	/<page-id>/feed	-	Page-ID; Access Token	Authorization	POST	Message	Post-ID
Get thermostats	/structures	Access Token	Access Token	Accept	GET	-	Thermostat-ID
Get target temperature	/devices/thermostats/<id>	Access Token	Access Token; Thermostat-ID	Accept	GET	-	Temperatur in Celsius
Set target temperature	/devices/thermostats/<id>	Access Token	Access Token; Thermostat-ID	Content-Type	PUT	Temperatur in Celsius	-
Get Devicelist	/devicelist	Access Token	Access Token	-	GET	-	Netatmo Geräte-IDs und Messwerte
Get Weather By Zipcode	-	zip; APPID	Zip-Code, Land-ID; App-ID	-	GET	-	Wetterdaten
Get Weather By Cityname	-	q; APPID	Stadtname; App-ID	-	GET	-	Wetterdaten
Get Light State	/<username>/lights/<id>	-	Username; Licht-ID	-	GET	-	Status, wie On/Off oder Lichtfarbe (Hue)
Set Light State	/<username>/lights/<id>/state	-	Username; Licht-ID	-	PUT	Status, wie On/Off oder Lichtfarbe (Hue)	-

Methode	*URL*	*Queries*	*Platzhalter*	*Header*	*Art*	*Request*	*Response*
Search for Artist	search?type=artist	q	Suchkriterium; Access Token	Authorization	GET	-	Name und URL zu Ergebnissen
Search for track	search?type=track	q	Suchkriterium; Access Token	Authorization	GET	-	Name und URL zu Ergebnissen
Get Playlist By Id	users/<user-id>/playlists/<playlist-id>	-	User-ID; Playlist-ID; Access Token	Authorization	GET	-	Bestimmte Playlist eines Nutzers
Get User Data	me	-	Access Token	Authorization	GET	-	Nutzerinformationen

Tabelle 2: Integration der Methoden

4.3. Integration der WunderBar

Da die WunderBar-API keine Methode anbietet, um Live-Daten per HTTP-Request anzufordern, muss die Einbindung per zusätzlichem SDK erfolgen.

Die Integration der WunderBar erfolgt über das Relayr-Java-SDK, das auf der Herstellerseite heruntergeladen werden kann. Durch dieses werden (wie bei der CloudRail-SDK) Calls an einen Endpunkt durchgeführt. Im Fall der WunderBar ist dieser Endpunkt die Cloud des Nutzers. Von dort holt sie sich die Sensormessungen, die im CloudRail-Büro in Mannheim gesammelt werden. Aus diesem Grund müssen externe Nutzer keine WunderBar besitzen, bzw. sich auf der Cloud einloggen. Die Anwendung verbindet sich automatisch über einen Token mit der Cloud. Das SDK wird im Projekt-Ordner `WebContent/WEB-INF/lib/` zusammen mit dem CloudRail-SDK platziert.

4.3.1. Auswertung durch SDKs

Die Wunderbar-SDK arbeitet mit verschiedenen Einheiten. Die erste Einheit ist die *user*-Klasse. Durch diese können später Daten des Nutzerkontos sowie verknüpfte Transmitter abgerufen werden. Ein *Transmitter*, die nächste Einheit, steht konkret für eine WunderBar. Der Entwickler hat also die Möglichkeit, mehrere WunderBars in die Anwendung zu implementieren. Unter den *Transmitter* fallen die *Devices,* welche für die einzelnen Sensoren der WunderBar stehen. Die Messungen der Sensoren werden als *Readings* bezeichnet.

4.3.2. Anbringen der Sensoren

Während der Testphase entstand das Problem, dass die WunderBar nicht die versprochenen 25 Metern Reichweite besitzt. Manche Sensoren konnten tatsächlich nur in kürzester Reichweite von maximal 20 Zentimeter Daten sammeln, ohne die Verbindung zum Hauptmodul abzubrechen. Daher musste das Hauptmodul mit dauerhafter Stromversorgung in unmittelbarer Nähe zu den Sensoren sein.

Abbildung 47: Befestigung des Näherungssensors

Der Näherungssensor wird dazu verwendet, die Menge an Kaffee pro Tag zu messen. Wie Abbildung 47 zeigt, sind auf der Kaffeemaschine zwei Knöpfe angebracht.

Der Näherungssensor erkennt jede Annäherung an die Sensoren. Je näher ein Objekt kommt, desto höhere Werte gibt er aus. Mit einem Referenzwert kann nun bei einer Schwellenüberschreitung der Zähler der Kaffeekachel hochgezählt werden.

Das Hauptmodul, der Temperatursensor und Lautstärkesensor wurde in unmittelbarer Nähe befestigt. Alle Module sitzen versteckt unter dem Boden des Regals.

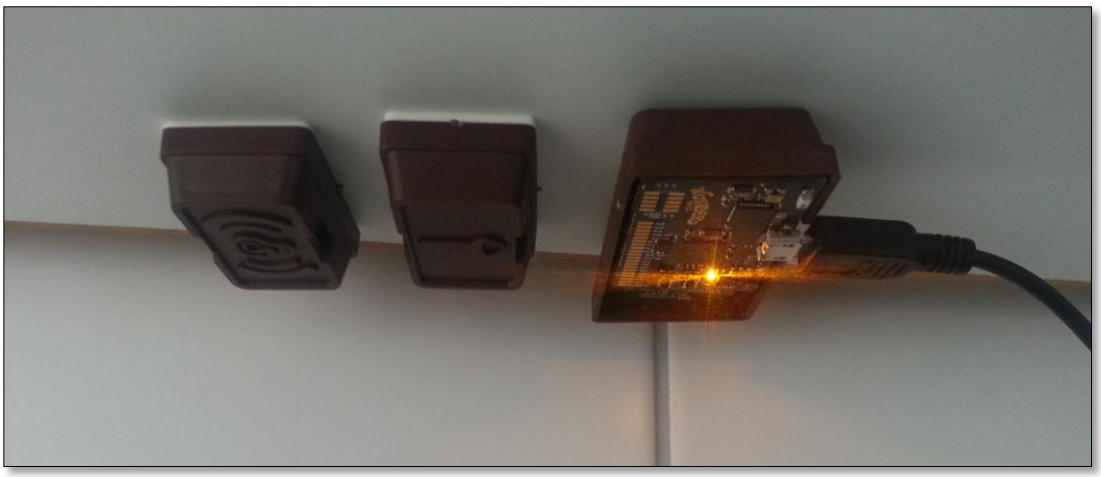


Abbildung 48: Befestigung der restlichen Module

Eine Plastikhülle schützt die Sensoren vor äußeren Einwirkungen. Am USB-Port des Hauptmoduls ist ein Stromladekabel befestigt, damit die Batterie nicht erschöpft.

Alle Dienste und Geräte sind in der CloudRail-Workbench definiert. Die WunderBar-SDK wurde integriert und die Daten aus der Cloud können abgerufen werden. Der nächste Schritt sieht die Implementierung des Dashboards vor.

4.4. Aufsetzen der Entwicklungsumgebung

Für die Umsetzung des Dashboards ist Vaadin ein hilfreiches Framework. Vaadin basiert auf Java und stellt eigene Bibliotheken und Designs zur Verfügung. Sobald ein Programm in Eclipse geschrieben ist, kann es als Web-Anwendung durch einen lokalen Server aufgerufen werden. Hierfür ist Tomcat von Nutzen. Im Folgenden wird der Aufbau dieser Entwicklungsumgebung kurz erläutert.

4.4.1. Java SDK

Das Java SDK[31] ist Voraussetzung für die Nutzung von Vaadin, sowie für die Eclipse IDE. Das SDK beinhaltet eine Laufzeitumgebung (Java Runtime Environment – JRE), einen Compiler und mehrere Bibliotheken, die im Entwicklungsprozess importiert werden können.

Das Java SDK lässt sich auf der Website von Oracle kostenlos herunterladen und einfach installieren.

[31] Java SDK steht für Java SE (Standard Edition) Development Kit

4.4.2. Apache Tomcat

Um fertige Projekte anzuschauen, reicht es in Java nicht, sie lokal im Browser zu öffnen. Die Verarbeitung durch einen Server ist nötig, um Actions und Events gleichermaßen auszutesten. Tomcat eignet sich als lokaler Web-Server für Entwicklungsaufgaben in Java. Auch Tomcat ist eine ausführbare Software, die kostenlos online erworben werden kann. Einmal eingerichtet, lässt sich der Server über ein Batch-File[32] starten. Über die URL `localhost:8080` kann nun die Startseite von Tomcat aufgerufen werden. Hier sind mehrere Beispiele und Dokumentationen zu finden.

4.4.3. Eclipse IDE

Eclipse ist mehr als ein Editor wie Notepad. Mit Eclipse kann eine Entwicklungsumgebung geschaffen werden, indem der Code geschrieben, kompiliert und über einen lokalen Server aufgerufen wird. Es können ganze Projekte erstellt werden, die über die Programmiersprache Java hinausgehen. Unterstützt wird auch PHP und C/C++. Eclipse steht auf der Herstellerseite zum kostenlosen Download zur Verfügung.

4.4.4. Vaadin

Als Plug-in für Eclipse dient Vaadin mit Bibliotheken und Designs zur Unterstützung des Entwicklers. Die Installation erfolgt direkt in Eclipse. Unter dem Reiter *Help* und *Install new software* wird im Feld *Work with* die URL `http://vaadin.com/eclipse` eingetragen. Eclipse stellt nun das Vaadin-Plug-in zur Installation zur Verfügung.

4.4.5. Erstellen eines Vaadin-Projektes

Um die Bibliotheken und Themes von Vaadin zu nutzen, ist das Erstellen eines Vaadin Projektes in Eclipse Voraussetzung. Die Auswahl steht nach der Installation des Plug-ins zur Verfügung. Vaadin generiert daraufhin den Projektordner, der im Gegensatz zum normalen Java-Projekt bereits über eine Testdatei verfügt, mehrere Bibliotheken enthält und eine Strukturierung innehat.

[32] Der Begriff Batch-File stammt aus DOS-Zeiten. Ein Batch-File führt eine Reihe von Befehlen aus. [75]

Die Abbildung 49 zeigt das Ergebnis. Links sind alle erstellten Projekte im *Project Explorer* aufgelistet.

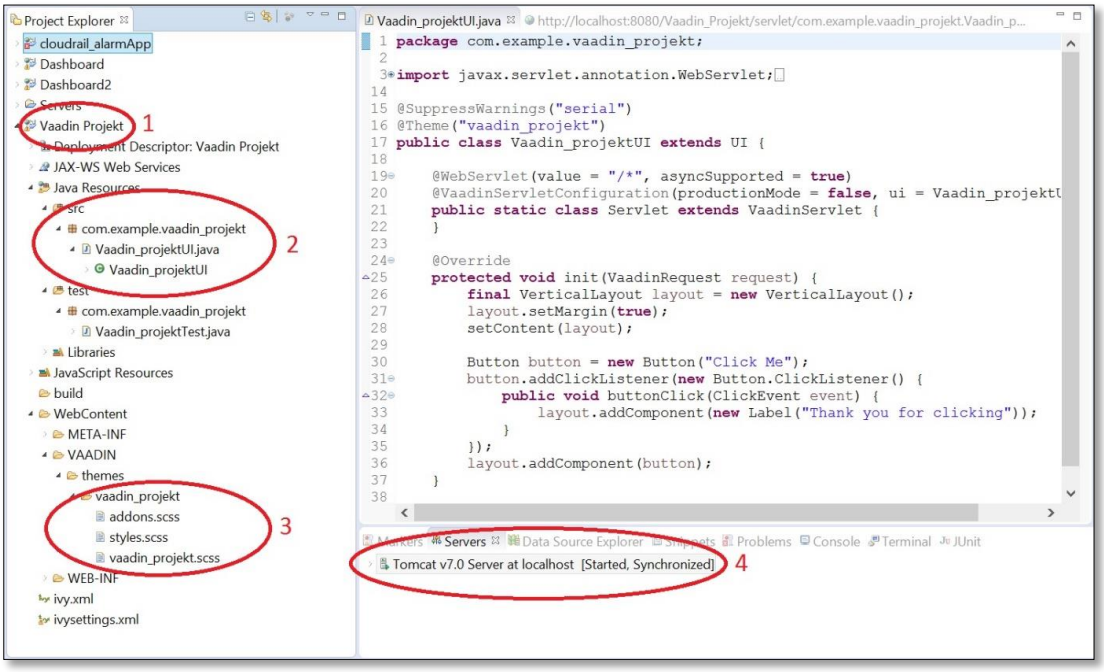


Abbildung 49: Vaadin Test-Projekt

1: Das erstellte Beispiel-Projekt heißt „Vaadin Projekt“. Die Namen sind frei wählbar.

2: Unter *Java Resources/src* ist eine Beispieldatei automatisch angelegt worden. Der Quellcode ist im Fenster rechts zu sehen.

3: Im Pfad *WebContent/Vaadin/themes/vaadin_projekt* sind mehrere .scss-Dateien angelegt worden. Diese sogenannten SASS-Dateien sind Metasprachen der Formatierungssprache CSS. Durch sie können demnach Formatierungen und Designs definiert werden.

4: Unter dem Reiter *Servers* wird der aktive Server angezeigt. In den Klammern steht der Status.

Um die Datei auf dem Server auszuführen, muss dieser erst gestartet werden. Entweder kann dies über das Batch-File erfolgen oder per Rechtsklick, *Start Server*.
Im Browser kann die Java-Anwendung nun über die entsprechende URL aufgerufen werden. Allgemein sieht diese wie folgt aus:

`http://localhost:8080/Projektname`

Für das Beispiel bedeutet dies:

`http://localhost:8080/Vaadin_ Projekt`

Abbildung 50: Ausgabe des angelegten Vaadin-Projektes

Der Code in Abbildung 49 liefert als Ergebnis einen Button, der bei Anklicken die Rückmeldung *Thank you for clicking* liefert. (siehe Abb. 50)

4.5. Erstellen des Dashboards

Bei der Programmierung in der Programmiersprache Java ist es empfehlenswert, den Code in möglichst viele Klassen aufzuteilen. Dies dient der Übersichtlichkeit und der einfacheren Konfiguration einzelner Bausteine. Java ist eine objektorientierte Sprache, was bedeutet, dass der Code in Klassen hierarchisch gegliedert werden kann. Klassen einer Ebene können in Packages[33] gebündelt werden.

Das Projekt hat folgende Gliederung inne:

- **dashboard**
  In diesem Package findet sich die DashboardUI-Klasse. Die Klasse initiiert das gesamte Dashboard. Für den Aufruf ist es also lediglich notwendig, diese Datei auf dem Server auszuführen.

- **page**
  Hier werden alle Seiten für das Ergebnis abgelegt. Da für mein Dashboard nur eine Seite vorgesehen ist, beinhaltet das Package lediglich die MainPage-Klasse. Diese ist für den Aufbau der Seite verantwortlich.

- **component**
  Im Komponenten-Package sind einzelne Bausteine hinterlegt. Bspw. sind die Sidebar, der Titel und die Zeitdarstellung in einer eigenen Klasse implementiert. Das Package soll dazu dienen, aufwendigere Bausteine aus dem Code herauszunehmen, um diesen übersichtlich zu halten.

- **overview**
  Im Overview-Package sind die einzelnen Kacheln vorhanden. Beispielklassen sind demnach die Facebook-Overview oder PhilipsHue-Overview. In den Klassen wird die Dashboard-Klasse im Interface-Package aufgerufen.

- **userInterface**
  Während die Overview-Klassen sich um die Oberfläche und das Design kümmern, ist in dem Interface-Package die durch CloudRail generierte Java-Klasse. Diese beinhaltet die Methoden, die auf der Workbench integriert wurden und Initiierungsmethoden für die Authentifizierung. Des Weiteren sind bereits Listener vordefiniert, die in die jeweilige Klasse importiert werden kann.

[33] dt. Pakete, eine Strukturierungsebene in Eclipse

In den folgenden Kapiteln wird jedes Package mit seiner Funktion im Gesamtkontext präsentiert.

4.5.1. Erstes Package - Dashboard

Fokussiertes Package: Dashboard
Beinhaltete Klassen: DashboardUI
Verknüpfte Packages: Page

Das erste Package beinhaltet lediglich eine Klasse. Das Package soll der Initiierung dienen und keine weiteren Aufgaben beinhalten. So kann später der Code über eine Klasse, bzw. ein Package ausgeführt werden.

4.5.1.1. Die Initiierung des Dashboard

Fokussierte Klasse: DashboardUI
Super-Klassen: UI
Interfaces: -

In dieser Klasse wird das gesamte Dashboard initiiert. Die Super-Klasse *UI* ist von Vaadin generiert und vererbt nützliche Methoden. Abgebildet ist der Code der einzigen Methode der Klasse.

```
protected void init(VaadinRequest request) {

    Responsive.makeResponsive(this);

    setContent(new MainPage());

    setSizeFull();}
}
```

Code 2: Die Initiierung des Dashboard

Die *init*-Methode besitzt keinen Rückgabewert (*void*). Der erste Ausdruck, der von der UI-Klasse vererbt wurde, verleiht dem Dashboard eine Anpassungsfähigkeit. Bei Veränderung der Größe des Browser-Fensters oder bei Wiedergabe auf verschieden großen Displays, passen sich die Werte der Höhe und Breite von nicht-fixen Elementen dem Fensterausschnitt an.
Der Methodenaufruf *setContent* legt den Inhalt fest. In diesem Fall ruft die Methode die Klasse *MainPage* auf. Zuletzt wird die UI auf Fenstergröße gesetzt. Dies bedeutet, dass die verbauten Elemente stets das Fenster ausfüllen.

4.5.2. Zweites Package - Page

Fokussiertes Package: Page
Beinhaltete Klassen: MainPage
Verknüpfte Packages: Dashboard, Overview, Component, UserInterface

Im zweiten Package wird vor allem der Aufbau der Seite beschrieben. Das erste Package, das das Dashboard komplett aufrufen soll, initiiert letztendlich nur das Page-Package. Dort werden die Kacheln (Overview), Komponenten (Component) und das CloudRail-SDK (UserInterface) instanziiert.

4.5.2.1. Der Aufbau der Seite

Fokussierte Klasse:	MainPage
Super-Klassen:	CustomComponent
Interfaces:	View, DashboardEditListener

Die *MainPage*-Klasse kümmert sich um den Aufbau der Dashboard-Seite. In der Klasse werden die Sidebar und Titel, sowie die Anordnung der Kacheln beschrieben.

Die Super-Klasse *CustomComponent* stammt aus dem Framework und vererbt einige Funktionen, wie *setCompositionRoot()*, die den Aufbau der Seite festlegt. Das Package der Klasse muss am Anfang des Codes importiert werden:

```
import com.vaadin.ui.CustomComponent;
```

Die Interfaces *View* und *DashboardEditListener* werden für die Umsetzung des Editier-Buttons benötigt. Mit diesem kann der Nutzer den Titel des Dashboards individuell ändern. Die Klasse View stammt aus dem Vaadin-Framework

```
import com.vaadin.navigator.View;
import com.vaadin.navigator.ViewChangeListener.ViewChangeEvent;
```

Der DashboardEdit-Button, sowie der zugehörige Listener werden in dem Package *component* implementiert und somit zu späterem Zeitpunkt erklärt.

```
import com.cloudrail.component.DashboardEdit;
import com.cloudrail.component.DashboardEdit.DashboardEditListener;
```

Weiterhin wird die Sidebar initiiert und im *Component*-Package aufgebaut. Im *MainContent* werden die Kacheln aufgerufen und platziert.

Der Aufbau der Seite erfolgt wie in Abbildung 51 dargestellt.

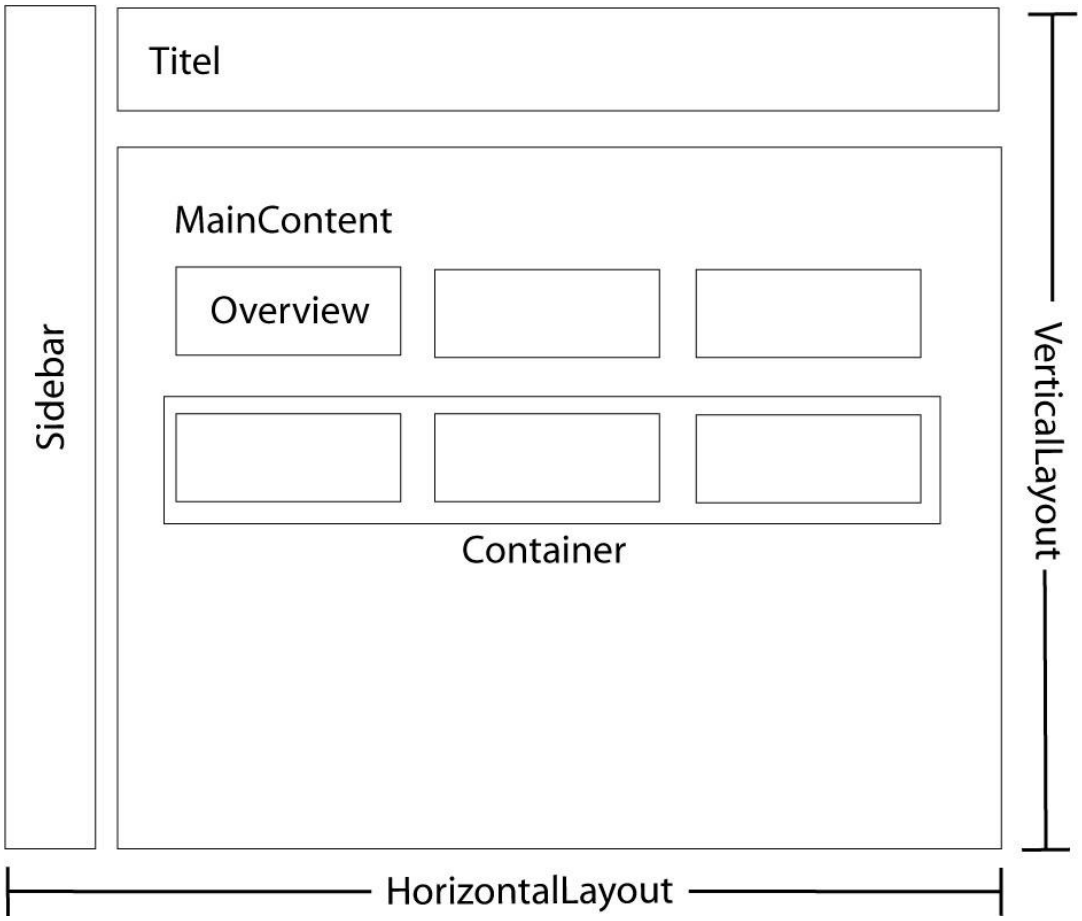


Abbildung 51: Aufbau des Dashboard

Zunächst wird ein horizontales Layout festgelegt. In dieses wird die *Sidebar* links und der *Content* rechts integriert. Der *Content* besteht wiederum aus einen vertikalen Layout, mit der Titelzeile oben und dem *MainContent* unten. Im *MainContent* werden die Kacheln durch die Overview-Klasse initiiert. Die Kacheln liegen in einem *Container*, der über ein CSS-Layout die Breite der Kacheln und deren Positionierung festlegt.

In Code 3 ist die Integration der Overviews der zweiten Reihe (*row2*) zu sehen.

```
private Component row2() {
        /* Integration Nest */
        Component nest = new NestOverview(dashboard);
        nest.addStyleName("small-spark");
        container.addComponent(nest);
        /* Integration OpenWeather */
        Component ow = new OpenWeatherOverview(dashboard);
        ow.addStyleName("small-spark");
        container.addComponent(ow);
        /* Integration Facebook */
        Component fb = new FacebookOverview(dashboard);
        fb.addStyleName("small-spark");
        container.addComponent(fb);

        return container;

}
```

Code 3: Integration der Overviews

Die Overviews werden durch eine Komponenteninstanz aufgerufen (siehe Code 3). Die Komponente erhält jeweils ein *StyleName*, sprich der Komponente wird ein Design hinzugefügt. Der Style *small-sparks* muss nun in einer der verknüpften .scss-Dateien definiert werden:

```
.small-sparks {
      position:relative;
       width: 30%;
       background: #002e62;}
```

Wie der Codeausschnitt zeigt, wird zunächst die Position der Kachel *relative* gesetzt. Dies hat den Nutzen, dass die Kachel sich nun nach der nächst höher gelegenen Bezugsebene positioniert (*container*). Durch die Angabe der Breite pro Kachel mit 30% wird bei 3 Kacheln erreicht, dass sie die Breite des *container* nicht komplett ausfüllen. Die letzten 10% stellen Ränder zwischen den Kacheln dar.

Jedem *Overview* wird die Instanz *dashboard* übergeben. Diese Instanz besitzt bereits die Werte der globalen Variablen aller Dienste, die einmal an den Graphen angehängt werden müssen und beim Aufruf des Dashboards ausgeführt wird. Globale Variablen sind im Beispiel von Facebook die App-ID, das App-Secret, die Redirect-URI, das CSRF-Token (State) und die Berechtigungen (Scope). Folgender Code zeigt die Instanziierung mit der Dashboard-Klasse aus dem SDK.

```
private Dashboard dashboard = new Dashboard(// globale
Variablen Nest, globale Variablen Facebook, globale Variablen
Netatmo,…);
```

4.5.3. Drittes Package - Component

Fokussiertes Package:	Component
Beinhaltete Klassen:	Sidebar, DashboardEdit, TimeClock
Verknüpfte Packages:	Page

Im dritten Package werden Komponenten ausgelagert, die einen längeren Quellcode aufweisen. So kann der eigentliche Code übersichtlich bleiben. Dazu zählen die Sidebar, der Editierungsbutton und die Zeitanzeige.

Aus Gründen der Übersichtlichkeit wird auf die Darstellung der Implementierung des Designkontextes verzichtet. Da das Package *component* zu großen Teilen solche Designelemente beinhaltet, werden die Klassen in Kürze beschrieben.

4.5.3.1. Sidebar

Fokussierte Klasse:	Sidebar
Super-Klassen:	CustomComponent
Interfaces:	-

Die Sidebar beinhaltet mehrere Links, eine Zeitanzeige und eine Kurzanleitung zum schnellen Einstieg in CloudRail.

4.5.3.2. Der Editierungsbutton

Fokussierte Klasse:	DashboardEdit
Super-Klassen:	Window
Interfaces:	-

Durch den DashboardEdit-Button können Nutzer den Titel des Dashboards individuell anpassen. Der Titel ist nur für den einzelnen Nutzer sichtbar und wird mit einem Seiten-Reload zurückgesetzt. Die Super-Klasse Window wird eingesetzt, um Methoden zur Anpassung des Edit-Windows zu vererben.

4.5.3.3. Zeitanzeige

Fokussierte Klasse:	TimeClock
Super-Klassen:	Label
Interfaces:	HasTimeHeaderListeners

Die Uhr ist Teil der Sidebar und aktualisiert sich nach der Systemzeit. Für das Dashboard bringt sie nicht nur einen gestalterischen Nutzen, sondern hilft auch bei der Implementierung der WunderBar. Um den Kaffee-Zähler täglich auf null zu setzen, wird eine einfache *if*-Bedingung implementiert.

```
if (formatter.format(currentTime).toString().equals("00:00:00")){
    File file1 = new File("prox.txt");
    try {FileWriter fw1 = new FileWriter("prox.txt");
    fw1.write(new Integer(0).toString());
    fw1.close();
          } catch (IOException e) {
          e.printStackTrace();}
```

Code 4: Zurücksetzen des Zählers durch die Uhr

Da lokale Variablen bei einem Refresh der Seite zurückgesetzt werden, muss für den Zähler der WunderBar eine externe Datei erzeugt werden (Darstellung in Kapitel 4.5.4.9).

Die aktuelle Uhrzeit wird mit 0 Uhr abgeglichen. Sobald dies wahr ist, wird der Zähler des Näherungssensors auf null gesetzt. Bei einem Refresh des Dashboards wird die Änderung dann wirksam. Auf diese Weise wird gewährleistet, dass der Zähler nur Messungen eines Tages summiert.

4.5.4. Viertes Package – Overview

Fokussiertes Package:	Overview
Beinhaltete Klassen:	DropboxOverview, FacebookOverview, HueOverview, NestOverview, NetatmoOverview, OpenWeatherOverview, SpotifyOverview, TemperatureHueOverview, WeatherMusic, WunderBarProx, WunderBarNoise, WunderBarTemperature
Verknüpfte Packages:	userInterface

Im Overview-Package werden alle Dienste integriert. Hierzu zählen das Erstellen der Kacheln und ihrer grafischen Oberfläche, das Einbinden des CloudRail-SDK (userInterface) und die Definition der Ein- und Ausgaben.

4.5.4.1. Facebook

Fokussierte Klasse:	FacebookOverview
Super-Klassen:	CustomComponent
Interfaces:	-

Um auf Facebook posten zu können, muss sich der Nutzer zunächst authentifizieren.

```
login.addClickListener(new ClickListener() {

public void buttonClick(ClickEvent event) {

    dashboard.init_FacebookGraphAPI_Auth(uricaller,
    uriRedirectListener, new AccessTokenCallback() {

    public void onToken(String at) {
          accessToken = at;
          Notification.show("You are authenticated. Share a
          message!", Type.TRAY_NOTIFICATION);
                        }
    @Override
    public void onError(CloudRailException arg0) {
    Notification.show("Error in authentication process",
    Type.TRAY_NOTIFICATION);}
    });
    dashboard.start_FacebookGraphAPI_Auth();}

});
```

Code 5: Facebook Integration

Der Button *login* bekommt einen *ClickListener*, welcher dafür benutzt wird, Klicks des Nutzers in Aktionen umzuwandeln. Klickt der Nutzer auf Login, wird der Authentifizierungsprozess gestartet.

Das Objekt *dashboard*, eine Instanz der durch die CloudRail-Workbench generierten SDK-Klasse *Dashboard*, initiiert die Authentifizierung. Die Methode *init_FacebookGraphAPI_Auth()* ist ebenso wie *start_FacebookGraphAPI_Auth()* Teil der SDK und muss lediglich aufgerufen werden. Die Authentifizierung erfolgt in diesen zwei Schritten und sieht bei jeder API ähnlich aus. Zu übergebende Parameter sind ein *localuricaller*, ein *uriRedirectListener* und ein *AccessTokenCallback*. Der *localuricaller* ist eine

Instanz der Klasse *LocalUriCaller*. Diese Klasse wird zu lokalen Testzwecken benötigt. Sie ist für das Öffnen jeder URI im lokalen Browser zuständig.

Der *uriRedirectListener* wird durch die Klasse *HttpsServerRedirectListener* instanziiert. Die Klasse benötigt einen *SSLContext* und einen Port, auf den später die Redirects durchgeführt werden können.

> *„Ein sogenannter SSL-Kontext stellt ein Gerüst für neue SSL-Verbindungen dar, indem er bestimmte Rahmenbedingungen für neue, am Kontext abgeleitete SSL-Verbindungen festschreibt."* [77]

Bestandteil des Kontextes sind unter anderem Informationen über den Inhaber und ein Passwort. Diese Parameter müssen beim Aufbau einer SSL-Verbindung nicht neu bestimmt werden, sondern werden im SSL-Kontext zusammengefasst und beim Anlegen neuer Verbindungen berücksichtigt. Der *Listener* wird letztlich dazu eingesetzt, im spezifizierten Port auf Redirects zu warten, um diese weiterleiten zu können.

Folgende Importe zeigen die Instanziierung beider Objekte.

```
final static LocalUriCaller localuricaller = new
LocalUriCaller();
final static UriRedirectListener uriRedirectListener = new
HttpsServerRedirectListener(Helper.getDummyContext(), 8081);
```

Aktionen, die durch die Authentifizierung ausgelöst werden sollen, können nun in die *onToken*-Methode eingefügt werden. In Code 5 wird der durch Facebook generierte Access Token in eine lokale Variable gespeichert. Mit dieser können nun die Posts durchgeführt werden.

Der Authentifizierungsprozess ist nicht mit dem Login zu verwechseln. Ist der Nutzer im Hintergrund bereits bei Facebook eingeloggt, wenn er sich auf dem Dashboard authentifiziert, wird dies erkannt und der Nutzer muss sich nicht erneut einloggen. Dementsprechend ist ein Logout auf der Anwendung nicht notwendig. Dies zeigt, dass das OAuth-Verfahren tatsächlich dazu führt, dass die externe Anwendung keine Daten des Nutzers verwalten kann und der Login über den Hersteller der API abgewickelt wird.

Code 6 auf der folgenden Seite zeigt wie eine Nachricht auf Facebook geteilt wird.

```
userPostSubmit.addClickListener(new ClickListener() {
    public void buttonClick(ClickEvent event) {

      if (accessToken == "") {
          Notification.show("Please authenticate yourself by
          clicking Login", Type.TRAY_NOTIFICATION);

    } else {
          dashboard.postAsUser(userInput.getValue(), accessToken,
          new PostAsUserResponseListener() {

                /* define actions on success */
                public void onSuccess() {
                Notification.show(userInput.getValue() + " was
                shared", Type.TRAY_NOTIFICATION);
                uricaller.open("https://www.facebook.com/me");}

                /* define actions on error */
                public void onError(CloudRailException error) {
                Notification.show("You have to insert a message",
                Type.TRAY_NOTIFICATION);
                getUI().push();

                public void onProgress(double percentFinished) {}
});}}
```

Code 6: Teilen eines Facebook-Post

Dem Button *userPostSubmit* wird ein *ClickListener* hinzugefügt. Im Beispiel-Code ruft das *dashboard*-Objekt die *postAsUser*-Methode auf, übergibt den *userInput* und den durch die Authentifizierung erlangten Access Token. Im *ResponseListener* kann wiederum definiert werden, was bei erfolgreichem (*onSuccess*) oder nicht erfolgreichem (*onError*) Verlauf des Calls und während der Wartezeit (*onProgress*) des Calls passieren soll.

Bei Erfolg wird eine Notiz ausgegeben, in der

```
My First Post! was shared.
```

stehen könnte.

Schließlich wird die Facebook-Seite des Nutzers geöffnet. Die Methode *getUI().push()* kann in jeder Rückgabemethode verwendet werden. Dadurch dass die API-Calls teilweise ein längeres Zeitintervall benötigen, müssen die in Rückgabemethoden definierten UI-Elemente *gepusht* werden. Ansonsten würde der Call ohne Reaktion auf der UI durchgeführt werden.

Für die Methode *postAsPage* wiederholt sich die Vorgehensweise. Lediglich die Seiten-ID muss als weiteres Objekt in dem Methodenaufruf übergeben werden.

Da der Authentifizierungsprozess bei allen APIs nahezu identisch ist, wird im Folgenden auf eine ausführliche Darstellung der Integration verzichtet und lediglich die Kerninhalte aus dem Code erläutert.

4.5.4.2. Philips Hue

Fokussierte Klasse:	HueOverview
Super-Klassen:	CustomComponent
Interfaces:	-

Die Methoden der Hue umfassen das Ein- und Ausschalten der Lampe, das Regeln der Helligkeit und Einstellen der Farbe.

Code 7 zeigt, wie die Lampe eingeschalten wird.

```
/* method to turn the light on/off */
lightButton.addClickListener(new ClickListener() {
    @Override
    public void buttonClick(ClickEvent event)  {
    try {
    /* isOn defines the current light state */
    if (isOn == false) {
    /* set the light state to on = true */
    dashboard.setLightStateOn(true, "lightswitch-v4", "1",new
    SetLightStateOnResponseListener() {
    public void onSuccess() {
          isOn = true;
          Notification.show("Light is on",
          Type.TRAY_NOTIFICATION);
```

Code 7: Einschalten der Hue-Lampe

Der Boolean *isOn* gibt den aktuellen Status an. Dementsprechend steht *false* für den ausgeschalteten und *true* für den eingeschalteten Zustand der Lampe.

Wenn der Nutzer nun auf den virtuellen Lichtschalter drückt, muss zunächst sichergestellt werden, dass die Lampe aus ist.

Dies geschieht über die *if*-Bedingung. Der *else*-Fall schaltet analog die Lampe aus. Ist also die Lampe ausgeschalten, wird die Methode *setLightStateOn* im SDK ausgeführt. Der erste Boolean gibt den gewünschten Lichtzustand an, danach folgen der Lampenname von CloudRail, die Lampen-ID und der *ResponseListener*.

Wie bereits dargelegt, besitzt jeder *ResponseListener* in der SDK die Rückgabemethoden *onSuccess*, *onError* und *onProgress*. Bei Erfolg wird also der Lampenzustand auf *true* gesetzt, es wird eine Notiz ausgegeben dass die Lampe erfolgreich eingeschalten wurde und das Icon auf der Kachel wird mit einer leuchtenden Glühbirne ersetzt. Außerdem wird eine weitere Methode aufgerufen.

```
dashboard.getLightState("lightswitch-v4", "1",new
GetLightStateResponseListener() {

@Override
public void onSuccess(Boolean state, Long brightness, Long
color,Long saturation) {
briValue = brightness;
hueValue = color;
satValue = saturation;}
```

Code 8: Abrufen der Lampenwerte

Da die Lampe beim Einschalten bereits über Werte verfügt, nämlich einer Farbe, Helligkeit und Sättigung (Code 8), müssen die Elemente der grafischen Oberfläche diese Werte annehmen. Zum Beispiel könnte der Nutzer den Helligkeitsregler bei ausgeschalteter Lampe verschieben. Würde die Lampe daraufhin eingeschaltet werden, hat sich ihre Helligkeit nicht verändert, obwohl der Regler dies anzeigt. Die Methode *getLightState* führt diesen Request durch. Der Name und die ID der Lampe werden übergeben. Als Response wird der Lampen-Zustand (*on/off*), die Helligkeit, Farbe und Sättigung vom Server gesendet. Diese Werte werden in lokalen Variablen gespeichert und für die Anpassung der Elemente verwendet.

4.5.4.3. Nest Thermostat

Fokussierte Klasse:	NestOverview
Super-Klassen:	CustomComponent
Interfaces:	-

Mit Nest kann der Nutzer seine aktuelle Thermostat-Temperatur abrufen und einstellen. Für beide Methoden müssen eine ID und eine Access Token übergeben werden. Die ID des Thermostats kann wiederum über eine Methode in Erfahrung gebracht werden. Um mit einem Knopfdruck die ID und die Temperatur eines Thermostats abzurufen, muss der Code wie folgt aufgebaut werden. In Code 9 ist das Abrufen der Nest-Temperatur implementiert.

```
/* method to obtain the current thermostat temperature */
getTemperature.addClickListener(new ClickListener() {
public void buttonClick(ClickEvent event) {
    if (accessToken == "") {
    Notification.show("Please authenticate yourself by clicking
    Login", Type.TRAY_NOTIFICATION);
    } else {
/* get all of your thermostats */
    dashboard.getThermostats(accessToken, new
    GetThermostatsResponseListener() {
          @Override
          public void onSuccess(List<Object> idList) {
/* get the temperature of one specific thermostat
insert any thermostat-id of your id-list*/
    dashboard.getTargetTemperature(idList.get(0).toString(),
    accessToken,new GetTargetTemperatureResponseListener() {
    @Override
    public void onSuccess(Double temperature) {
    System.out.println(temperature);}
```

Code 9: Abrufen der Nest-Temperatur

Durch den Button *getTemperature* wird die Temperatur abgerufen. Der Button prüft zunächst über eine *if*-Bedingung, ob der User bereits einen Access Token über den Autorisierungsprozess generiert hat. Die Autorisierung wird wie bei Facebook über einen Login-Button abgewickelt. Ist ein Access Token vorhanden, wird die Methode *getThermostats* aufgerufen. Diese übergibt bei erfolgreicher Anfrage eine Liste mit IDs (*idList*), wobei jede ID für ein Thermostat steht. Mit der ID kann nun die Temperatur abgefragt werden. Im Beispiel wird die erste ID aus der Liste zusammen mit dem Access Token übergeben. Die Antwort des Servers ist die Temperatur des mit der ID übereinstimmenden Gerätes. Diese wird in einen String geschrieben und ausgegeben.

Eine beispielhafte Antwort wäre also:

```
Your thermostat temperature amounts to 21.5°C
```

Für die Durchführung der Temperaturänderung (setTemperature) muss lediglich eine Temperatur im Request übergeben werden. Der Nutzer trägt also seine neue Wunsch-Temperatur ein und kann über die *getTemperature*-Methode die erfolgreiche Durchführung überprüfen.

4.5.4.4. Netatmo Weatherstation

Fokussierte Klasse:	NetAtmoOverview
Super-Klassen:	CustomComponent
Interfaces:	-

In der Kachel zum Netatmo-Dienst ist ein einziger Button integriert. Mit diesem Button kann das Raumklima abgerufen werden. Der Autorisierungsprozess ist dabei mit der Methodendurchführung gekoppelt. Bei Abfrage der Messungen autorisiert sich der Nutzer also automatisch, ohne vorher mit einem zusätzlichen Button diesen Prozess aufrufen zu müssen. Der Vorteil hierbei ist, dass der Ablauf benutzerfreundlicher ist, auch wenn die Autorisierung mit jedem Call durchgeführt werden muss.

Der Ablauf wurde auf diese Weise gewählt, da der Nutzer sein Raumklima nicht permanent einsehen wird. Dies steht im Gegensatz zu dem Prozess in Facebook oder Nest, bei dem der Nutzer zuerst den Login durchführt und dann ohne weitere Autorisierung Methoden durchführen kann. Hierbei wird die Autorisierung mit jeder Aktion sehr schnell störend. Um die Prozesse der Autorisierung und Klimaabfrage zu koppeln, muss der Code wie abgebildet aufgebaut werden.

```
/* method to obtain the room climate */
getMeasure.addClickListener(new ClickListener() {
    @Override
    public void buttonClick(ClickEvent event) {
    /* initiate authorization process */
    dashboard.init_NetatmoWeatherstationAPI_Auth(uricaller,
    uriRedirectListener,new AccessTokenCallback() {
    public void onToken(String accessToken) {
    /* get your netatmo devices */
    dashboard.getAtmoDevices(accessToken, new
    GetAtmoDevicesResponseListener() {
    @Override
    /* all relevant data is transferred in one Array*/
    public void onSuccess(String status, List<Object> idList) {
    /* print your results, you should sort them first*/
```

Code 10: Abrufen des Netatmo-Raumklimas

Der Button *getMeasure* initiiert zunächst die Autorisierung. Wie schon mehrfach dargelegt, werden der *localuricaller* und *uriRedirectListener* als Parameter übergeben. In der Rückgabemethode *onToken* wird nun der *accessToken* als Parameter an die Methode *getAtmoDevices* übergeben. Diese gibt alle in der Cloud registrierten Netatmo-Geräte mit Messungen in einer Liste an die Anwendung zurück. Die Liste, bzw. der Array kann dann gefiltert und sortiert ausgegeben werden.

4.5.4.5. OpenWeather

Fokussierte Klasse:	OpenWeatherOverview
Super-Klassen:	CustomComponent
Interfaces:	-

Durch die OpenWeather-Kachel können aktuelle Wetterdaten eines Ortes abgerufen werden. Die Anfrage kann entweder über die Eingabe eines Stadtnamens oder der Postleitzahl (engl. zip code) und dem dazugehörigen Ländercode erfolgen. Dadurch dass einige Städte den gleichen Namen haben, ist diese Art der Anfrage genauer.

In Code 11 wird das Wetter durch die Eingabe der Postleitzahl abgerufen.

```
/* Get weather data by zip code */
zipSubmit.addClickListener(new ClickListener() {

    @Override
    public void buttonClick(ClickEvent event) {

/* call method with User inserted parameters zipInput,
countryCodeInput and the apiKey generated on OpenWeather.com*/
    dashboard.getWeatherByZip(Double.parseDouble(zipInput.getValue
    ()), countryCodeInput.getValue(), apiKey,new
    GetWeatherByZipResponseListener() {

    /* get the outputs specified on the CloudRail workbench */
          public void onSuccess(String cityname, Double
          temperature, Double pressure, Long humidity,Double
          temperature_min, Double temperature_max, Double
          wind_speed, Double wind_direction,
          Long cloudiness, List<Object> condition) {

          String outcome = "My weather data for " + cityname + ":
          The thermostat shows " + temperature+ " degrees Celsius,
          humidity is about " + humidity + "% and the sky is
          filled with "+ cloudiness + "% clouds. Minimum
          temperature today were " + temperature_min+ " degrees
          Celsius and maximum temperature reached " +
          temperature_max+ " degrees Celsius. Wind blows at a
          speed of " + wind_speed + " meters per second.";
          getUI().push();}

          public void onError(CloudRailException error) {
          Notification.show("Something went wrong. Check the zip-
          code and try again!",Type.TRAY_NOTIFICATION);
          getUI().push();}

          public void onProgress(double percentFinished) {}
```

Code 11: Abfrage des Wetters durch die Postleitzahl

Durch den Button *zipSubmit* wird die Anfrage mit der Postleitzahl und dem Ländercode als Parameter durchgeführt. Die ausführende Methode wurde *getWeatherByZip* genannt. Als Parameter werden der vom User eingegebene zip code (*zipInput),* der Ländercode (*countryCodeInput)*, ein auf OpenWeather generierter API-Key (*apiKey*) und der Response-Listener übergeben.

Bei erfolgreichem Durchlauf werden die auf der Workbench festgelegten Parameter zurückgegeben. Dazu zählen neben Temperatur und Luftfeuchtigkeit auch Windgeschwindigkeit und Grad der Bewölkung. Die Parameter werden in einen String geschrieben und ausgegeben. Eine beispielhafte Response wäre demnach:

```
My weather data for Offenburg: The thermostat shows 23.2
degrees Celsius, humidity is about 23% and the sky is filled
with 44% clouds. Minimum temperature today were 12.4 degrees
Celsius and maximum temperature reached 24.1 degrees Celsius.
Wind blows at a speed of 4.5 meters per second.
```

Dieser kurze Report mit einem Knopfdruck auf Facebook gepostet werden. Die Idee dabei ist nicht, nur Wetterdaten zu teilen, sondern auch zu sehen, von wo auf der Welt Nutzer auf das Dashboard zugreifen.

4.5.4.6. Spotify

Fokussierte Klasse:	SpotifyOverview
Super-Klassen:	CustomComponent
Interfaces:	-

Um Spotify nutzen zu können, müssen sich Nutzer zunächst einloggen. Die Spotify API bietet lediglich Funktionen, die eine Authentifizierung voraussetzen. Da die Nutzer möglicherweise mehrfach nach Liedtiteln oder Künstlern suchen, ist eine mit den Methoden gekoppelte Authentifizierung benutzerunfreundlich. Mit jedem Request würde zunächst der Authentifizierungsprozess durchlaufen werden. Darum empfiehlt sich ein Login-Button, mit dem sich Nutzer zunächst authentifizieren. Wie bei der Implementierung von Nest und Facebook wird der generierte Access Token in einer lokalen Variable gespeichert. Nun können Methoden mit dem Access Token aufgerufen werden.

Folgend wird detailliert die Methode zum Suchen nach Künstlern beschrieben.

```
artistSubmit.addClickListener(new ClickListener() {
    public void buttonClick(ClickEvent event) {
    if (accessToken == "") {
          Notification.show("Please authenticate yourself by
          clicking Login", Type.TRAY_NOTIFICATION);
    } else {
          String artist = artistInput.getValue();
          String result = "";
    for (int i = 0; i < artist.length(); i++) {
          char hiChar = artist.charAt(i);
          if (hiChar == ' ') {
          hiChar = '+';}
          result += hiChar;}
```

Code 12: Künstlersuche

Die Suche nach einem Künstler umfasst ein Textfeld (*artistInput*), in das der Künstler eingetragen wird und ein Button (*artistSubmit*) zum Abschicken des Request.

Zuerst wird wieder überprüft, ob durch das Authentifizieren des Nutzers in den Access Token ein Wert geschrieben wurde. Wenn dem so ist, muss der Eintrag des Nutzers modifiziert werden. Da bei Suchfeldern Leerfelder möglich sind, sprich der Nutzer nach „The Beatles“ suchen könnte, müssen diese durch ein „+“ ersetzt werden, um eine gültige Anfrage zu schaffen. Zur Erinnerung, das Suchkriterium wird später als Query an die URL angehängt und darf daher keine Leerfelder aufweisen. Um dies zu erreichen, wird der Eintrag des Nutzers durchlaufen. Jeder Char-Wert[34] wird nun mit einem Leerfeld verglichen und bei Übereinstimmung durch das „+“ ersetzt.

[34] Ein char ist eine Instanz ähnlich String oder Integer. Während Strings Zeichenketten definieren und Integer ganze Zahlenwerte, stehen chars für einzelne characters, zu Deutsch Buchstaben.

Code 13 zeigt die Durchführung des Request. Das in der Variable result vom Nutzer eingegebene und modifizierte Suchkriterium wird zusammen mit dem Access Token als Parameter übertragen. Als Response sendet der Server einen Array mit den Antworten auf die Anfrage. Zunächst wird nun überprüft ob dieser Array überhaupt über Inhalte verfügt. Ist er leer, wird eine Notiz ausgegeben. Beinhaltet er Werte, können diese geordnet ausgegeben werden.

```
/*method to obtain artist search results*/
    dashboard.searchforartists(result, accessToken, new
    SearchforartistsResponseListener() {

    public void onSuccess(List<Object> items) {
    /*checking if result is empty*/
    if (items.isEmpty()) {
    Notification.show("No results found!",
    Type.TRAY_NOTIFICATION);
    } else {
    /*print items*/
    System.out.println(items);
```

Code 13: Auswertung der Künstlersuche

Zu den Rückgabewerten zählt der Name des Künstlers, ein Profilbild und ein Link auf die Spotify-Seite des Künstlers.

4.5.4.7. Visualisierung der Ortstemperatur

Fokussierte Klasse:	TemperatureHueOverview
Super-Klassen:	CustomComponent
Interfaces:	-

In dieser Kachel kann der Nutzer seine Hue-Daten und seine Heimatstadt eingeben um dann eine Farbänderung auf der CloudRail-Hue zu erzielen, die sich nach den Außentemperaturen anpasst. Zunächst wird also die Temperatur der Stadt durch die OpenWeather-API (*getWeatherByCity)* abgerufen. Dann folgt ein Algorithmus, der die Temperatur kategorisiert und je nach Gradzahl die Werte der Helligkeit, Farbe und Sättigung bestimmt. Die Methode *setLightStateOnHueBriSat* schaltet die Lampe ein und führt dann die Änderungen auf der Lampe aus (siehe Code 13).

```
/*method to change the hue state to true with predefined color-,
saturation- and brightness-values*/
dashboard.setLightStateOnBriHueSat(true, bri, color, sat,
"lightswitch-v4","1", new
SetLightStateOnBriHueSatResponseListener()
{@Override

public void onSuccess() {
    Notification.show(temperature + "°C in " + cityname + ". Light
    color customized.",Type.TRAY_NOTIFICATION);}
@Override
public void onError(CloudRailException error) {
    Notification.show("Error occurred. Check your inputs and try
    again!",Type.TRAY_NOTIFICATION);}
@Override
public void onProgress(double percentFinished) {}
```

Code 14: Wertänderungen auf der Hue-Lampe

Bei erfolgreichem Durchlauf, wird eine Notiz mit der angefragten Temperatur gezeigt:

`24.7°C in Palma de Mallorca. Light color customized.`

Dieses Beispiel zeigt, wie einfach es ist, die Aktionen der verschiedenen Dienste miteinander zu kombinieren. Dadurch dass sie über die CloudRail-Schnittstelle miteinander vernetzt sind, müssen lediglich die Calls definiert werden.

4.5.4.8. Playlist für lokale Temperatur

Der Nutzer hat hier die Möglichkeit, eine Spotify-Playlist für seine örtliche Temperatur vorgeschlagen zu bekommen. Dies bedeutet, dass bei kalten Werten zu winterlichen und bei warmen Temperaturen zu sommerlichen Playlists zurückgegeben wird.

```
/* initiate authorization process */
    dashboard.init_SpotifyWebAPI_Auth(uricaller,
    uriRedirectListener,new AccessTokenCallback() {

    public void onToken(String accessToken) {
    /* the OpenWeather method is already done, the temperature
    leads to a playlist with predefined spotify- and playlistIDs*/
    dashboard.getPlaylist(spotifyId, playlistId, accessToken,new
    GetPlaylistResponseListener() {

    public void onSuccess(String url, String description, String
    name){
    Notification.show("Playlist " +name+"fits to
    your"+cityname+"'s temperature of "+temperature+"°C");
```

Code 15: Abruf einer speziellen Playlist

Code 15 zeigt, wie eine Playlist auf Spotify abgerufen werden kann. Zunächst müssen jedoch die *spotifyId* und *playlistId* festgelegt werden. Ersteres ist die ID des Erstellers der Playlist. Der Request für den Abruf von Ortstemperaturen durch die Eingabe eines Stadtnamens wurde bereits gezeigt. Der Rückgabewert der Temperatur wird nun in Bedingungen aufgeteilt. Ist die Temperatur beispielsweise kleiner oder gleich 0 Grad Celsius, so wird die Playlist „Acoustic Winter“ des Nutzers „spotify“ abgerufen.

Für den Aufruf müssen also nur die beiden Werte zusammen mit dem Access Token übergeben werden. Zurück kommen eine URL, eine Beschreibung und der Name der Playlist. Die URL führt zum Web-Player von Spotify, durch den die Playlist direkt angehört werden kann. Als Notiz wird ausgegeben:

`Playlist ACOUSTIC WINTER fits to Offenburg's local temperature of -2.5°C.`

Die weiteren Rückgabemethoden sind bereits bekannt und wurden deshalb aus dem Codeausschnitt genommen.

4.5.4.9. WunderBar - Näherungssensor

Fokussierte Klasse:	WunderBarProx
Super-Klassen:	CustomComponent
Interfaces:	-

Der Näherungssensor der WunderBar wurde in der Küche an der Kaffeemaschine angebracht (siehe Kapitel 4.3.2). Sobald die Maschine betätigt wird, registriert der Sensor die Näherung und zählt den Kaffeezähler hoch.

Zunächst muss aber das relayr SDK im Projekt initialisiert werden.

```
new RelayrJavaSdk.Builder().setToken("Bearer
<Token>").build();
```

Die RelayrJavaSDK-Klasse ist Teil des SDK. Die Methode *Builder()* initialisiert das SDK. In den Token-Platzhalter wird der entsprechende Token des Nutzers eingetragen. Dieser kann auf der Herstellerwebsite generiert werden. *build* baut schließlich die Verbindung zur Cloud auf.

```
RelayrJavaSdk.getUser().subscribe(new Action1<User>() {
     public void call(User user) {
               System.out.println(user);
               start(user);}});
```

Im nächsten Schritt muss der *user* instanziiert werden. Der dargestellte Code liest den Nutzer aus. Die Verbindung zum Nutzerkonto erfolgt über den in der Initiierung übergebenen Token. Die Instanz der User-Klasse wird nun der *start*-Methode übergeben

In der *start*-Methode werden die Transmitter des Users aufgerufen.

```
user.getTransmitters().flatMap(new Func1<List<Transmitter>,
Observable<List<TransmitterDevice>>>(){}.subscribe();
```

Die Methode *subscribe* wird verwendet, um an die Daten des Transmitter zu gelangen. Wie in der CloudRail-SDK bereits kennengelernt, hat sie drei Rückgabemethoden.

- ***onCompleted***
  In der Methode können Aktionen für den abgeschlossenen Durchlauf einer Transmitteranbindung definiert werden. Die Methode kann nicht für das Darstellen von Live-Daten benutzt werden, da der Vorgang zu dem Zeitpunkt abgeschlossen wird und keine Daten mehr gesendet werden.
- ***onError***
  Bei Fehlern im Prozess, können hier Fehlermeldungen für den Nutzer eingetragen werden.
- ***onNext***
  Die Methode liefert als Rückgabeparameter eine Liste von *Devices*. Mit dieser Methode können also im Weiteren die *Readings* ausgelesen werden.

Die folgenden Zeilen zeigen, wie zunächst der Sensor ausgewählt wird und dessen Messungen dann abgerufen werden. Zunächst wird die Liste von *Devices* durchlaufen bis der *Device* mit der ID des Näherungssensors erreicht wurde. Dieser wird nun der Methode *subscribeToReadings* übergeben, um die Messungen auszulesen.

```
for (TransmitterDevice device : devices)
if (device.getId().equals("d0c3-4f90-8583-eb76aa8d722a")) {
subscribeToReadings(device);}
```

Auch *subscribeToReadings* hat die drei Rückgabemethoden *onCompleted*, *onError* und *onNext*. Letztere bekommt die Messung als *Reading*-Instanz übergeben. Diese Instanz ist ein Objekt mit mehreren Variablen. Interessant für die Auswertung sind *meaning* (Beschreibung des Sensors) und *value* (der tatsächliche Wert).

```
if (reading.meaning.equals("proximity")) {
if ((double) reading.value > 200.0) {}
```

Obwohl bereits die *device-ID* auf den Näherungssensor festgelegt wurde, muss nun erneut eine Spezifizierung des *meaning* erfolgen. Der Näherungssensor kann nicht nur herankommende Objekte identifizieren, sondern auch das Licht messen `(reading.meaning.equals(„light"))` und Farben scannen `(reading.meaning.equals(„color"))`. Mit 200.0 wird nun ein Referenzwert festgelegt, welcher bei Überschreitung eine Aktion auslöst. Diese Aktion besteht darin, dass ein Zähler hochgesetzt wird und an das Dashboard ausgegeben wird. Auf diese Weise wird eine Näherung an den Sensor auf dem Dashboard visualisiert.

Die anderen Sensoren werden analog ausgegeben. Die ausführliche Darstellung im Code ist dem Anhang zu entnehmen.

4.5.4.10. WunderBar - Thermostat

Fokussierte Klasse:	WunderBarTemperature
Super-Klassen:	CustomComponent
Interfaces:	-

Die Kachel zeigt die aktuelle Temperatur, die der Sensor misst. Im Gegensatz zum Näherungssensor muss die ID des Devices und die Bedeutung der Messung geändert werden `(reading.meaning.equals(„temperature"))` für die Temperatur und `(reading.meaning.equals(„humidity"))` für die Luftfeuchtigkeit.

Die Temperatur kann nun als Kachel-Überschrift angezeigt werden.

4.5.4.11. WunderBar - Lautstärkesensor

Fokussierte Klasse:	WunderBarNoise
Super-Klassen:	CustomComponent
Interfaces:	-

Über den Sensor wird die aktuelle Lautstärke in der Küche ausgegeben. Leider konnte der Sensor aus Gründen der Reichweitenbeschränkung zum Hauptmodul nicht im Büro angebracht werden. Die Lautstärke wird darüber hinaus nicht in Dezibel gemessen, sondern in einer von relayr erfundenem Messsystem.

Die Kachel gibt je nach Lautstärkelevel ein Bild aus, das diesen Level visualisiert und zeigt die Differenz zur vorherigen Messung an.

4.5.5. Das CloudRail Interface

Fokussiertes Package:	userInterface
Beinhaltete Klassen:	Dashboard, CloudRail Konfigurationsdatei
Verknüpfte Packages:	Overview

Im Interface-Package (dt. Schnittstelle) werden die Dateien aus der Zip-Datei, die von der Workbench heruntergeladen wurde, eingefügt.

4.5.5.1. Das CloudRail SDK

Fokussierte Klasse:	Dashboard
Super-Klassen:	-
Interfaces:	-

Die Dashboard-Klasse ist durch die Einstellungen auf der Workbench automatisch generiert worden.

Der Klasse werden alle globale Variablen von der *MainPage* übergeben und gesetzt.

```
globals.put("philipshueapi_ipaddress",philipshueapi_ ipaddress);
```

Der Endpunkt wird dann über die globalen Variablen und die Konfigurationsdatei erstellt.

```
endpoint = new Endpoint (getClass().getResourceAsStream
("Dashboard.cloudrail"), globals);
```

Die Klasse beinhaltet außerdem alle in der Workbench definierten Methoden und Rückgabewerte. Auch die *Response-Listener* sind als Interface integriert.

```
public void postAsUser(String message, String accesstoken, final
PostAsUserResponseListener answer) {

HashMap<String, Object> inputs = new HashMap<String, Object>();
    inputs.put("message", message);
    inputs.put("accesstoken", accesstoken);

endpoint.call("postAsUser", inputs, new
EndpointCallResponseListener() {

public void onSuccess(Map<String, Object> result) {
          answer.onSuccess();}
public void onError(CloudRailException error) {
          answer.onError(error);}
public void onProgress(double percentFinished) {
          answer.onProgress(percentFinished);}
    });
}
```

Code 16: Beispiel eines Interfaces aus dem CloudRail-SDK

Code 16 zeigt eine solche Methode. Dargestellt ist das Teilen von Inhalten an die eigene Pinnwand (*postAsUser*). Der Methode werden eine Nachricht (*message*) sowie der im Autorisierungsprozess generierte Access Token übergeben. Im Code sah der Methodenaufruf so aus:

```
dashboard.postAsUser(userInput.getValue(), accessToken, new
PostAsUserResponseListener()
```

Die *message* ist der eingegebene Text des Users und der Access Token wurde durch das Drücken des Login-Buttons erstellt (siehe Kapitel 4.5.4.1).

Die Methode im SDK nimmt nun die Inputs und führt unter dem Namen der auf der Workbench definierten Methode (*postAsUser*) den Endpunkt auf. Bestandteil des Aufrufs ist auch der *EndpointCallResponseListener*, der die Rückgabewerte übergibt.

Bei erfolgreichem Durchlauf wird beim Teilen eines Postes kein Rückgabewert gegeben, da die optionalen Parameter der *User_ID* und *Post_ID*, mit denen Facebook auf einen Post antwortet, auf der Workbench nicht als erwünschte Werte angekreuzt wurden.

4.5.5.2. CloudRail Konfigurationsdatei

In Kapitel 2.5.3 wurde bereits beschrieben, wie CloudRail das Interoperabilitätsproblem löst. Zusammenfassend werden die vom Nutzer eingetragenen Definitionen in Graphen übertragen. Je nach Muster werden spezifische Plugins aktiviert, die die HTTP Requests auslösen.

Während das SDK mit zur Verfügung stehenden Methoden und Interfaces den Entwickler aktiv unterstützt, wird die Konfigurationsdatei passiv in den Projektordner integriert. In der Datei sind die Knoten der Graphen definiert. Diese wurden aus der Action-Definition auf der Workbench abgeleitet.

```
"1.9C$39.1": "facebookgraphapi_clientidentifier",
"1.A2$39.1": "facebookgraphapi_uri",
"1.A8$39.1": "facebookgraphapi_scope",
"1.AE$39.1": "facebookgraphapi_clientsecret",
"1.B4$39.1": "facebookgraphapi_csrftoken",
"1.B5$39.1": "Facebook Graph API",
"1.C6$39.1": "message",
"1.D3$39.1": "accesstoken"
```

Code 17: Beispiel aus der Konfigurationsdatei

Der abgebildete Code 17 zeigt einen Ausschnitt aus der Konfigurationsdatei. Den Platzhaltern und Methoden wird ein Knoten zugeordnet. Dabei werden die globalen Platzhalter zuerst an den Graphen angehängt. Hierzu zählen der *clientidentifier*, *uri*, *scope*, *clientsecret* und *csrftoken*. Hierzu als kurze Erinnerung die definierten globalen Variablen von Facebook aus Kapitel 4.2: App-ID, Redirect-URI, Scope, App-Secret, State.

Der Name ist in den Actions frei wählbar und kann deshalb von den Namen in der API-Definition abweichen. Wichtig ist, dass die Werte dem jeweiligen Parameter zugeordnet werden. Nach den globalen Variablen folgt die Methode, die durch den API-Namen ersetzt ist. Unter der Methode folgen die lokalen Variablen *message*, *pageIdentifier* und *accesstoken*. Hiermit kann die Methode *postAsPage* durchgeführt werden.

4.6. Upload auf den CloudRail-Server

Bevor das Dashboard auf einem Server ausgeführt werden kann, müssen wenige Modifikationen durchgeführt werden. Das von CloudRail generierte SDK stellt einen SSL-Kontext und einen URI-Caller zur Verfügung, die beide nur auf einem lokalen Rechner funktionieren. Um die Anwendung auch auf dem Server ausführen zu können, müssen diese beiden Klassen angepasst werden.

URI-Caller

Der *LocalUriCaller* aus dem SDK öffnet eine Seite im lokalen Browser. Dies bedeutet, dass die Klasse auf dem Server versuchen würde, einen Browser zu finden und zu öffnen. Die Klasse ist also nicht funktionsfähig.

Es wird ein *UriCaller* benötigt, der die URL im Browser des Nutzers öffnet. Hierzu wird eine neue Klasse angelegt und mit einer entsprechenden Methode versehen.

```
public void open(String url) {
Page.getCurrent().open(new   ExternalResource(url),   "_blank",
true);}
```

Die Methode *open* in der neuen Klasse *UriCaller* führt eine übergebene *url* im aktuellen Browser des Nutzers aus. Um die Methode zu nutzen, muss eine Instanz der Klasse aufgerufen werden.

```
final static UriView uricaller = new UriCaller();
```

Der Aufruf der Methode wird über die instanziierte Variable ausgeführt.

```
uricaller.open(url);
```

Auf diese Weise wird erreicht, dass die Anwendung eine Seite im geöffneten Browser des Nutzers ausführt.

SSL-Kontext

Teil des SDK ist auch ein SSL-Kontext, der für lokale Testzwecke verwendet werden kann. Um auch Redirects auf dem Server funktionsfähig zu machen, muss dieser SSL-Kontext für den Server angelegt werden.

Der neue Kontext beinhaltet die SSL-Strukturen des Servers, ein neues Passwort und die angepassten IP-Adressen. In der Klasse *SSL* wird die Datei in der Methode *getSSL* aufgerufen und der Kontext ausgelesen.

Die Instanziierung des *uriRedirectListener* muss nun wie folgt aussehen.

```
final static UriRedirectListener uriRedirectListener = new
HttpsServerRedirectListener(SSL.getSSL(),8083);
```

Der SSL-Kontext, der erste Parameter in der Instanz, wird nun nicht mehr aus dem SDK aufgerufen, sondern aus der eigens angelegten *SSL*-Klasse.

Aufsetzen der Anwendung auf dem Server

An dieser Stelle zeigen sich die Vorzüge von einer Entwicklungsumgebung wie Eclipse. Das gesamte Projekt kann verpackt und als WAR-Datei exportiert werden. In diesem Archiv befinden sich alle angelegen Java-Klassen, Bilder, XML-Dateien und weitere Ressourcen, die in der Erstellung der Anwendung erzeugt wurden.

Die WAR-Datei muss letztlich nur noch auf dem Server aufgesetzt werden. Auf Tomcat-Servern erfolgt dies wie in Abbildung 52 dargestellt.

Die WAR-Datei muss ausgewählt werden und wird schließlich mit „Deploy“ auf den Server aufgespielt.

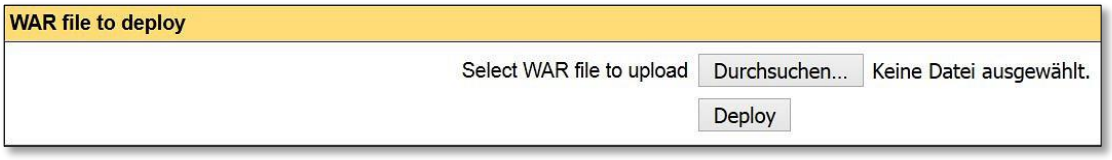


Abbildung 52: Hochladen der WAR-Datei auf den Server

4.7. Präsentation des Dashboards

Jeder Dienst wurde nun auf das Dashboard integriert. Das Dashboard ist funktionsbereit und wurde auf der Homepage hochgeladen. Auf den folgenden Seiten wird ein kurzer Überblick gegeben.

Die Sidebar
Um Nutzern einen schnellen Anschluss an CloudRail und weitere Infos zu geben, sind in der Sidebar einige Links eingefügt. Über den Titel *CloudRAIL* lässt sich die Homepage aufrufen. *Workbench* öffnet dementsprechend die Startseite der Workbench. Unter Quick Start wurde ein Fenster implementiert, das in drei Schritten erklärt wie die erste Anwendung mit CloudRail gelingt. Die Schritte „Visit our Workbench“, „Save the SDK“ und „Integrate the SDK into your code“ werden visualisiert nacheinander ausgegeben. *About us* öffnet die entsprechende Seite auf der Homepage.

Am Ende der Sidebar finden sich das Datum und die aktuelle Zeitanzeige, die sich sekündlich aktualisiert. Darüber sind die Web-Auftritte von CloudRail verlinkt. Dazu zählen Facebook, LinkedIn, YouTube, Twitter und Google+.

Der Hauptteil
Wie im Aufbau beschrieben, findet sich ein editierbarer Titel am Anfang der Seite, gefolgt von den angeordneten Kacheln. Jede Kachel verfügt über ein eigenes Fenster, in dem die Funktionen ausgetestet werden können. Abbildung 54 auf der Folgeseite zeigt dieses Fenster am Beispiel von Spotify.

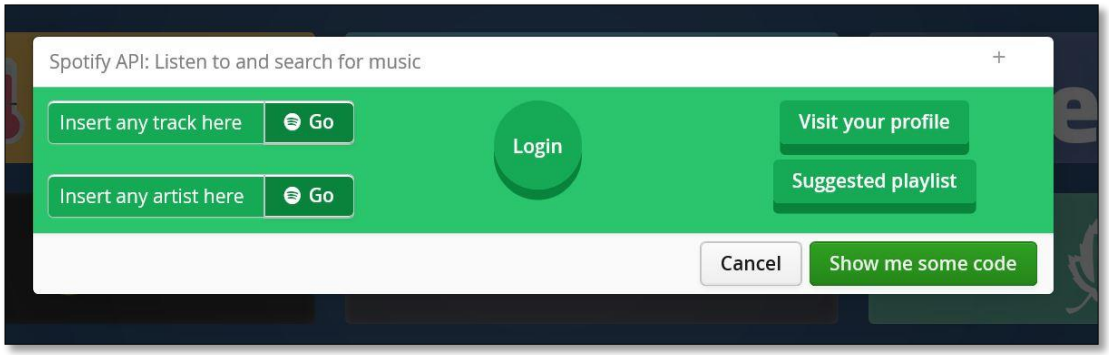


Abbildung 53: Das Spotify-Window

Abbildung 54: Das Ergebnis-Window von Spotify

Nach der Suche eines Songtitels, öffnet sich das Ergebnisfenster (siehe Abb. 55).

Als Ergebnis werden der Name des vollständigen Titels aus dem Musikkatalog, ein Bild des zugehörigen Albums, ein Ausschnitt aus dem Stück und ein Link zum Album zurückgegeben.

Das Dashboard ist kein Endprodukt, sondern kann stetig mit weiteren APIs erweitert und kombiniert werden, damit mehr Nutzer profitieren.

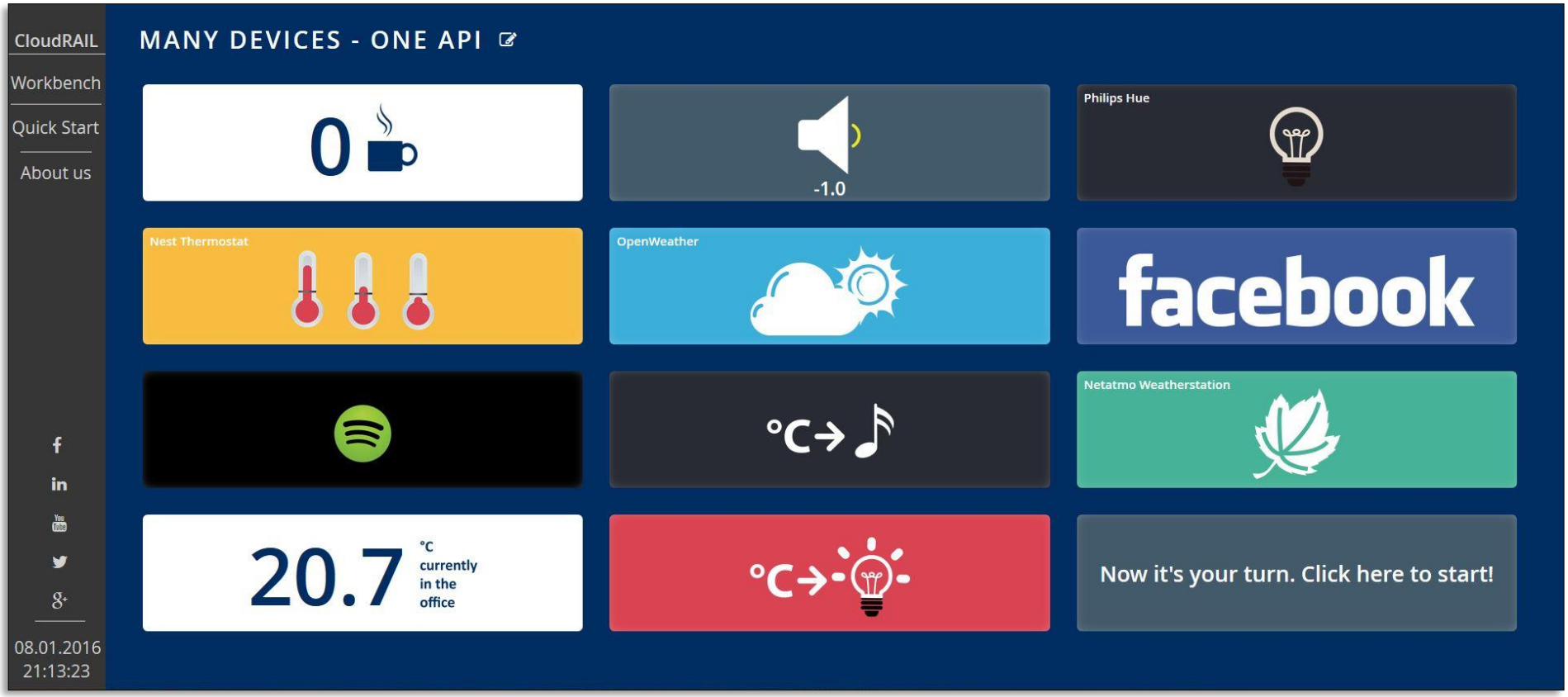


Abbildung 55: Das fertige Dashboard

4.8. Aufgetretene Probleme

Bei der Erstellung des Dashboards gab es teilweise große Verzögerungen. Folgend möchte ich diese erläutern.

Um der Aufgabenstellung gerecht zu werden, war es zunächst schwierig, einerseits passende Geräte zu finden, die durch eine Web-API angesteuert werden können und andererseits möglichst viele Nutzer anzusprechen. Einige Kacheln funktionieren leider nur für Nutzer, die das entsprechende Gerät Zuhause haben.

In der Testphase entstand ein Problem, das den Redirect auf den lokalen Tomcat-Server nicht mehr möglich machte. So konnten die APIs, zu deren Nutzung eine Authentifizierung benötigt wird, für einen langen Zeitraum nicht getestet werden.

Die WunderBar sollte ursprünglich auch in der CloudRail API definiert werden und die Live-Auswertungen über die API-Calls durchgeführt werden. Die WunderBar API bietet einige Funktionen, wie das Erstellen und Löschen eines Devices, gibt aber leider keine Daten über HTTP Requests preis. So musste ein zweites SDK integriert werden.

Schließlich möchte ich hervorheben, dass mir bei jedem Problem geholfen und ich damit nicht alleine gelassen wurde. Das Entwickler-Team nahm sich gerne Zeit, um meine Fragen zu beantworten und half mir darüber hinaus durch ausführliche Erklärungen bei der Erstellung dieser Thesis.

Kapitel 5 Zusammenfassung

Der erste Schritt bei der Erstellung dieser Arbeit war die Recherche zum Internet der Dinge. Im ersten Kapitel wurde gezeigt, wie bedeutend das IoT für uns in ein paar Jahren sein wird. Sei es in der Industrie oder im Eigenheim, vernetzte Geräte im IoT übernehmen Aufgaben und Funktionen im Alltag. Grundlegend hierfür sind APIs, Schnittstellen zwischen verschiedenen Systemen, die mit der wachsenden Zahl an Anwendungen immer wichtiger werden. Der Aufbau von RESTful Web Services und die Grundlagen der Web-APIs waren dabei Themen, die zunächst zu durchdringen waren. Die große Bandbreite an Möglichkeiten von Web-APIs und die ausführlichen Dokumentationen der Anbieter machten das Themengebiet leichter zu verstehen.

Folgend wurde gezeigt, dass Entwicklungsprozesse im IoT durch CloudRail stark vereinfacht werden. Während die Einigung unter Firmen auf einen Standard bis zum heutigen Tag fruchtlos war, bietet CloudRail eine standardlose Lösung für das Interoperabilitätsproblem. Die Nutzung des CloudRail-Systems und der Workbench war dabei schnell verstanden. Durch das Anfertigen eigener API Definitionen mit den Vorkenntnissen zu HTTP Requests und Responses sowie den Aufbau eines kleinen Testprojektes zum Testen der APIs, wurde das Verständnis der Funktionsweise immer besser. In der ersten Phase wurde also die Datenbank von CloudRail mit API-Definitionen gefüllt, um später auf eine größere Auswahl an Funktionen zurückgreifen zu können und eine konsistente Datenbank zu besitzen. Um Nutzern ein Beispiel zu geben, was mit CloudRail realisiert werden kann, wurde unter Zuhilfenahme der CloudRail-API ein Dashboard entworfen, das die Vernetzung verschiedener Geräte und Dienste visualisiert. Hierzu war ein Einlernen in das Vaadin-Framework nötig, was durch ein Buch erleichtert wurde. Mit dem Framework wurde ein Programmiergerüst des Dashboards entwickelt, in das die APIs integriert wurden. Diese mussten teilweise in der CloudRail-Workbench definiert werden. Während Geräte wie Netatmo und Nest bereits vorhanden waren, wurde die Workbench um weitere APIs wie Facebook, Spotify Web und OpenWeather ergänzt.

Abschließend stand das Design. Für viele Funktionen, wie die Steuerung der Hue-Lampe, war das Design der anspruchsvollste Part, da der Nutzer über selbstsprechende Buttons die Lampe regeln soll.

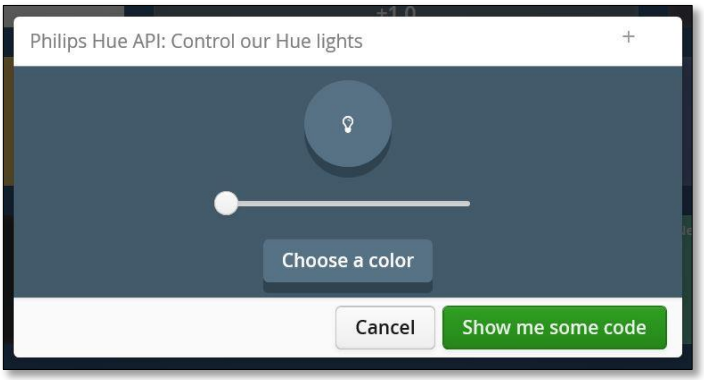


Abbildung 56: Philips Hue Steuerung

In Abbildung 56 ist das Ergebnis zu sehen. Die Lampe lässt sich über den Button oben in der Mitte ein- und ausschalten. Der Slider dient der Anpassung der Helligkeit. Unter „Choose a color“ können Farben ausgewählt werden.

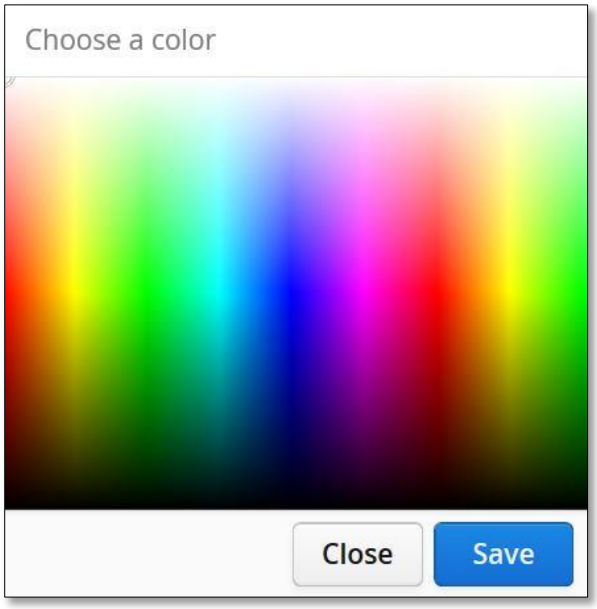


Abbildung 57: Das Color-Window der Hue-Kachel

Nebenstehende Abbildung 57 veranschaulicht die breite Farbauswahl, auf die der Nutzer zur Einstellung seiner Philips Hue zurückgreifen kann. Die Schwierigkeit war, die Werte als RGB an die API zu übertragen und korrekt darzustellen.

Die Farbpalette ist zwar vollständig vorhanden, die Philips Hue kann allerdings nicht den gesamten Farbraum darstellen. So sind Farbtöne im Schwarz- oder Weißbereich nicht anwählbar.

Bei anderen Kacheln, wie Spotify, war die Implementierung der Funktion sehr zeitintensiv. Der Nutzer kann beispielsweise Lieder durch ein Suchfeld finden. Das Auslesen der JSON-Response war hier problematisch. Spotify sendet eine längere Response, in der eine Fülle von Informationen, wie Bildern, Playlist-Links, Namen und IDs sowie Ausschnitte aus dem Song vorhanden sind. Diese Response war in mehrere Arrays und Objekte untergliedert, sodass das Auslesen von einzelnen Informationen schwierig wurde, vor allem unter dem Aspekt, dass die Bildauflösung je nach Popularität des Künstlers stark schwankte. Der technische Aufbau und Informationen zur Implementierung der Dienste in der CloudRail-Schnittstelle sowie deren Integration in das Dashboard wurden offengelegt. Besucher des Dashboards können beispielsweise ihre Smart-Home-Geräte steuern oder

Abbildung 58: Code-Window von Netatmo

Web-Anwendungen nutzen. Entwickler haben mit dem Dashboard die Möglichkeit, eigene Anwendungen in CloudRail mit Hilfe des in den Kacheln offen gelegten Codeszu

Abbildung 58 zeigt ein solches Code-Window. Jede Kachel, ausgenommen die der WunderBar, verfügt über einen kurzen Einblick in die Integration von CloudRail in die Anwendung.

5.2. Fazit

Der Technologietrend zeigt in die Richtung einer umfassend informatisierten Welt. Der Prozess der Digitalisierung passiert nicht über Nacht, vielmehr ist es eine schleichende Revolution. Diese Revolution betrifft dabei immer mehr Lebensbereiche, in der die reale Welt mit der virtuellen verschmilzt. Das IoT hat somit Einfluss auf jeden. Die heute einseitige Kommunikation mit Geräten wird zu einem Dialog ausgebaut. Die allgegenwärtige Vernetzung versorgt uns mit gewünschten Informationen und gibt solche über uns preis – überall und zu jeder Zeit.

Während in den Anfangszeiten des WWW befürchtet wurde, dass die Realität verschwindet und Computer die Oberhand gewinnen, kann man heute feststellen, dass Geräte im IoT benutzerzentriert arbeiten. Die virtuelle Welt richtet sich unsichtbar in unser aller Leben ein. Zeit und Arbeit sind Faktoren, die auf diese Weise eingespart werden können. Der Mensch kann sich auf die wirklich wichtigen Dinge im Leben konzentrieren.

Im Rahmen dieser Arbeit wurden einige Erkenntnisse über die Bandbreite an Möglichkeiten gewonnen. Das Dashboard steht beispielhaft für die Fülle an Diensten, die sich in einem Netz der Dinge bedienen und steuern lassen. Das IoT öffnet Türen für Entwicklungen, die vorher nicht realisierbar waren. Die Vernetzung von Diensten mit verschiedenen Funktionen, die sich gegenseitig ansprechen und steuern können, macht einen neuen Markt zugänglich. Firmen füllen diesen mit Produkten und viele Start-Ups versuchen ihn mit Softwares, Geräten und innovativen Ideen zu erschließen. Es hat aber den Anschein, als wäre den Firmen dabei egal, was der Mensch wirklich braucht. Dies bestätigen diverse Umfragen zum IoT, die auch in dieser Arbeit aufgezeigt wurden. Das IoT ist für Privatpersonen ein schönes Spielzeug, das mit einem teuren Preis einhergeht – dem Verlust der Privatsphäre. Spätestens seit der NSA-Affäre ist dem Endnutzer eine Naivität gegenüber der Verwaltung seiner Daten abzuschreiben und verschafft der Zukunft des IoT somit einen schweren Stand. Für den Nutzer müssen Fragen offen geklärt werden, um Akzeptanz und Vertrauen in das System entstehen zu lassen.

5.3. Interpretation und Bewertung

Was bedeutet es, wenn der Computer als physisches Gerät verschwindet und im Hintergrund agiert? Wie trennen wir zukünftig die reale von der virtuellen Welt, wenn selbst meine Kaffeemaschine an ein Netz von Geräten angeschlossen ist, die zur selbst- und gegenseitigen Steuerung befähigt sind? Und spielt der Datenschutz hierbei überhaupt noch eine Rolle?

Diese Fragen beschäftigten mich unter anderem in der Zeit der Recherche und Erstellung dieser Arbeit. Eine Antwort darauf konnte ich nicht finden. Ein Meinung allerdings schon:

Das IoT steht trotz zahlreicher, verfügbarer Produkte erst am Anfang seiner Entwicklung. Wenn wir uns erinnern, dass gerade einmal vor 25 Jahren der Siegeszug des WWW begann, indem wir erste Texte über das Internet schickten, gefolgt von Bildern und Videos, wir vor 10 Jahren anfingen, Menschen zu vernetzen und dies nun auf reale, zur Kommunikation unfähige Objekte erweitern, kann uns noch nicht bewusst sein, welche Folgen diese allgegenwärtige Vernetzung für uns haben wird.

Digitale Abhängigkeit ist ein Aspekt, der sich als ein massives Problem herauskristallisieren könnte. Schon heute scheinen sich viele Menschen nicht mehr ohne Handy zurechtzufinden. In der Zukunft weitet sich diese Abhängigkeit auf weitere virtualisierte Geräte aus.

Während gegenwärtig Gefahren aus dem Netz auf das Internet beschränkt sind, müssen zukünftig Sicherungen für jegliche Geräte in meinem Haushalt installiert werden. Denn eine allgegenwärtige Vernetzung führt zu allgegenwärtigem Sammeln von Daten. Dass Firmen mit diesen adäquat umgehen, mag seit der Sammelklage gegen Facebook wegen Weitergabe der Daten an US-Überwachungszentren und externen Anwendungen, keiner mehr glauben.[35]

Es sind viele Fragen zu klären. Recherchiert man das IoT, bemerkt man aber schnell, dass keine andere Technologie die Branche mehr beflügelt. Die Chancen scheinen die Risiken zu überbieten. Es bleibt zu hoffen, dass Politik und Wirtschaft den Datenschutz und den Menschen als zentralen Akteur nicht vernachlässigen.

[35] mehr Infos: `http://www.europe-v-facebook.org/DE/Anzeigen/Sammelklage/sammelklage.html`

Literaturverzeichnis

[1] *Das Internet der Dinge – oder: Wie alles vernetzt wird.* Verfügbar unter: http://www.giga.de/software/internet/das-internet-der-dinge-oder-wie-alles-vernetzt-wird/ (14.10.2015).

[2] *RWE SmartHome – Android-Apps auf Google Play.* Verfügbar unter: https://play.google.com/store/apps/details?id=com.rwe.it.sk.smarthome&hl=de (13.11.2015).

[3] T. Barton, *E-Business mit Cloud Computing: Grundlagen, Praktische Anwendungen, verständliche Lösungsansätze*. Wiesbaden: Springer Vieweg, 2014.

[4] T. Kaufmann, *Geschäftsmodelle in Industrie 4.0 und dem Internet der Dinge: Der Weg vom Anspruch in die Wirklichkeit*. Wiesbaden: Springer Vieweg, 2015.

[5] *Mojio, the connected car platform Raises $8M - [Jcount.com]*. Verfügbar unter: http://www.jcount.com/mojio-the-connected-car-platform-raises-8m/ (14.11.2015).

[6] *Botanicalls » Kits.* Verfügbar unter: http://www.botanicalls.com/kits/ (14.11.2015).

[7] W. electronic, *Arduino Uno R3 - Watterott electronic.* Verfügbar unter: http://www.watterott.com/de/Arduino-Uno (14.11.2015).

[8] K. Terplan und C. Voigt, *Cloud Computing,* 1. Aufl. Datacom-Buchverlag GmbH, 2011.

[9] *Dashboards & KPIs - Prospettiva - Business Software Solutions.* Verfügbar unter: http://prospettiva.co.uk/dashboards_kpis/ (14.11.2015).

[10] *Polar Loop.* Verfügbar unter: http://www.fitnessconcept.com.my/products/accessories/heart-rate-monitor/polar-loop-detail (22.10.2015).

[11] *OneDrive Dev Center - OneDrive authentication and sign-in.* Verfügbar unter: https://dev.onedrive.com/auth/msa_oauth.htm (26.10.2015).

[12] *Facebook Login for Apps - Developer Documentation - Dokumente - Facebook für Entwickler.* Verfügbar unter: https://developers.facebook.com/docs/facebook-login/permissions/v2.5 (16.10.2015).

[13] *Graph API - Dokumente - Facebook für Entwickler.* Verfügbar unter: https://developers.facebook.com/docs/plugins/like-button (28.10.2015).

[14] *CloudRail.* Verfügbar unter: https://angel.co/cloudrail (14.11.2015).

[15] *Felix Kollmar.* Verfügbar unter: https://www.f6s.com/felixkollmar (14.11.2015).

[16] *The IoT Standards Battle | SevOne.* Verfügbar unter: https://www.sevone.com/blog/iot-standards-battle (14.11.2015).

[17] licobo GmbH, *CloudRail Workbench.* Verfügbar unter: https://workbench.cloudrail.com/#/ (13.11.2015).

[18] *Intelligente Heizkörper Thermostate & Heizungsregler.* Verfügbar unter: http://www.heizsparer.de/heizung/heizkorper/thermostate/intelligente-thermostate (09.10.2015).

[19] *Bosch und das Internet der Dinge.* Verfügbar unter: https://www.bosch-si.com/de/internet-der-dinge/iot-bosch/iot-bosch.html (13.10.2015).

[20] M. Weiser, *The computer for the 21st century.*

[21] rbaheti, http://ieeecss.org/sites/ieeecss.org/files/documents/IoCT-Part3-02CyberphysicalSystems.pdf.

[22] *FAQ zum Internet der Dinge.* Verfügbar unter: http://www.internet-der-dinge.de/de/potenziale/faq-zum-internet-der-dinge.html (14.10.2015).

[23] F. Online, *Sorgen um Daten bei Fitnessarmbändern und vernetzter Heizung*. Verfügbar unter: http://www.focus.de/finanzen/news/wirtschaftsticker/umfrage-sorgen-um-daten-bei-fitnessarmbaendern-und-vernetzter-heizung_id_4841628.html (14.10.2015).
[24] *Smarthome Erlebniswelt - Geräte*. Verfügbar unter: https://www.rwe-smarthome.de/web/cms/de/2851388/home/geraete/ (14.10.2015).
[25] *Bluetooth Low Energy / Bluetooth Smart (4.0 / 4.1 / 4.2)*. Verfügbar unter: http://www.elektronik-kompendium.de/sites/kom/1805171.htm (13.11.2015).
[26] C. Metzger, *WLAN und Bluetooth: Gemeinsames und Unterschiede*. Verfügbar unter: http://www.pcwelt.de/ratgeber/WLAN-und-Bluetooth-Gemeinsames-und-Unterschiede-200723.html (13.11.2015).
[27] N. Jurran und c't, *Funkprotokoll ZigBee will sich neu erfinden*. Verfügbar unter: http://www.heise.de/ct/ausgabe/2015-1-Funkprotokoll-ZigBee-will-sich-neu-erfinden-2483856.html (13.11.2015).
[28] *Richard David Precht Industrie 4.0*. Verfügbar unter: https://www.youtube.com/watch?v=lZrlZ3Bgajo (20.10.2015).
[29] *Telekom-Chef Höttges für bedingungsloses Grundeinkommen*. Verfügbar unter: http://www.zeit.de/wirtschaft/2015-12/digitale-revolution-telekom-timotheus-hoettges-interview (30.12.2015).
[30] *How It Works*. Verfügbar unter: https://www.moj.io/how-it-works/ (27.10.2015).
[31] M. Miller, *The Internet of things: How smart TVs, smart cars, smart homes, and smart cities are changing the world*. Indianapolis, Indiana: Que, 2015.
[32] *Google Self-Driving Car Project*. Verfügbar unter: https://www.google.com/selfdrivingcar/faq/#q1 (26.10.2015).
[33] T. Brühlmann, *Arduino: Praxiseinstieg ; [behandelt Arduino 1.0],* 1. Aufl. Heidelberg, Hamburg: mitp Verl.-Gruppe Hüthig Jehle Rehm, 2012.
[34] h. online, *Raspberry Pi: Mini-Comupter mit ARM-Prozessor*. Verfügbar unter: http://www.heise.de/thema/Raspberry-Pi (22.10.2015).
[35] *AWS-Fallbeispiel: Netflix*. Verfügbar unter: https://aws.amazon.com/de/solutions/case-studies/netflix/ (29.10.2015).
[36] *Internet Of Things - ThingSpeak*. Verfügbar unter: https://thingspeak.com/ (14.11.2015).
[37] F. Mattern, Hg., *Total vernetzt: Szenarien einer informatisierten Welt ; 7. Berliner Kolloquium der Gottlieb Daimler- und Karl Benz-Stiftung*. Berlin: Springer, 2003.
[38] *Polar Loop - Aktivitätstracking mit smarter Anleitung | Polar Deutschland*. Verfügbar unter: http://www.polar.com/de/produkte/werde_aktiv/fitness/loop (27.10.2015).
[39] web&mobile Developer, "Windows 10 für Entwickler: Smartphone, Tablet oder Desktop: Mit der Universal Windows Plattform deckt Microsoft alle Gerätekategorien ab." 2015. Jg., 11/15, S. 36.
[40] Frankfurter Allgemeine Zeitung GmbH, *Protokoll einer Zukunftsvision: Das System versagt*. Verfügbar unter: http://www.faz.net/aktuell/feuilleton/debatten/kapitalismus/protokoll-einer-zukunftsvision-das-system-versagt-12057446.html (15.10.2015).
[41] *Internet of Things: Opportunities and Challenges - OpenMind*. Verfügbar unter: https://www.bbvaopenmind.com/en/internet-of-things-opportunities-and-challenges/ (20.10.2015).
[42] *What is Open? A simple description of APIs*. Verfügbar unter: https://www.youtube.com/watch?v=7r7QpIDEI_o (15.10.2015).

[43] *Web APIs for more than just Facebook, Twitter*. Verfügbar unter: http://searchsoa.techtarget.com/news/2240114646/Web-APIs-for-more-than-just-Facebook-Twitter (29.10.2015).
[44] E. Gamma, R. Helm, R. Johnson und J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*: Pearson Education, 1994.
[45] *IoC: inversion of control :: Umkehrung des Kontrollflusses :: ITWissen.info*. Verfügbar unter: http://www.itwissen.info/definition/lexikon/IoC-inversion-of-control-Umkehrung-des-Kontrollflusses.html (14.11.2015).
[46] *Need for APIs*. Verfügbar unter: https://developer.ibm.com/apimanagement/docs/api-101/need-api/ (14.10.2015).
[47] C. Anderson, *Makers: Das Internet der Dinge: die nächste industrielle Revolution,* 1. Aufl. München: Carl Hanser Fachbuchverlag, 2013.
[48] *At 15 Billion A Day, Twitter Sees 50% More API Calls Than Facebook & Google Combined [INFOGRAPHIC]*. Verfügbar unter: http://www.adweek.com/socialtimes/api-billionaires-club/452908?red=at (14.10.2015).
[49] *History of APIs*. Verfügbar unter: http://history.apievangelist.com/ (14.10.2015).
[50] *APIs Dashboard*. Verfügbar unter: http://www.programmableweb.com/apis (14.10.2015).
[51] N. Bessis, F. Xhafa, D. Varvarigou, R. Hill und M. Li, Hg., *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence*. Berlin, Heidelberg: Springer, 2013.
[52] S. Tilkov, *REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien,* 1. Aufl. Heidelberg: dpunkt, 2009.
[53] T. Lodderstedt, J. Hiller und h. Developer, *Flexible und sichere Internetdienste mit OAuth 2.0*. Verfügbar unter: http://www.heise.de/developer/artikel/Flexible-und-sichere-Internetdienste-mit-OAuth-2-0-2068404.html (22.10.2015).
[54] *HTTPS / HTTP Secure (HTTP + SSL/TLS)*. Verfügbar unter: http://www.elektronik-kompendium.de/sites/net/1811281.htm (29.10.2015).
[55] *About - One to Many API - Connect to Everything - CloudRail*. Verfügbar unter: https://cloudrail.com/about-cloudrail/ (14.10.2015).
[56] *About Us | Industrial Internet Consortium*. Verfügbar unter: http://www.iiconsortium.org/about-us.htm (31.10.2015).
[57] *About Us | Open Interconnect Consortium*. Verfügbar unter: http://openinterconnect.org/about/ (31.10.2015).
[58] K. Glahn und h. open, *AllJoyn: Freies P2P-Framework von Qualcomm*. Verfügbar unter: http://www.heise.de/open/meldung/AllJoyn-Freies-P2P-Framework-von-Qualcomm-1192911.html (28.10.2015).
[59] T. Group, *About*. Verfügbar unter: http://threadgroup.org/About.aspx (31.10.2015).
[60] *What is UPnP? » UPnP Forum*. Verfügbar unter: http://upnp.org/about/what-is-upnp/ (28.10.2015).
[61] H. Rügheimer, *Heimvernetzung: Das kann DLNA - connect*. Verfügbar unter: http://www.connect.de/ratgeber/heimvernetzung-das-kann-dlna-1471222.html (28.10.2015).
[62] *The Walled Garden Problem*. Verfügbar unter: https://www.artsjournal.com/rwx/2011/03/the-walled-garden-problem/ (28.10.2015).

[63] *Sid Lee Paris Dashboard*. Verfügbar unter: http://dashboard.sidlee.com/ (26.10.2015).
[64] *QuickTickets Dashboard*. Verfügbar unter: https://demo.vaadin.com/dashboard/#!dashboard (26.10.2015).
[65] *Philips Pulls Defective 1.7.1 Hue App Update After Crashing Reports [Update: Fixed]*. Verfügbar unter: http://www.macrumors.com/2015/02/06/philips-hue-app-crashing/ (14.11.2015).
[66] *Ambieye*. Verfügbar unter: http://ambieye.com/ (26.12.2015).
[67] *NEST Thermostat | Energy Circle*. Verfügbar unter: http://www.energycircle.com/shop/nest-thermostat (14.11.2015).
[68] N. Labs, *Nest Learning Thermostat*. Verfügbar unter: https://store.nest.com/product/thermostat/ (28.10.2015).
[69] *Edit Product*. Verfügbar unter: https://developer.nest.com/products/d9a6e183-a986-4773-83da-784c7c508e16/edit (05.12.2015).
[70] *Netatmo NWS01 Wetterstation für Apple iPhone/iPad/iPod und Android auf conrad.de bestellen | 000615765*. Verfügbar unter: https://www.conrad.de/de/netatmo-nws01-wetterstation-fuer-apple-iphoneipadipod-und-android-615765.html (14.11.2015).
[71] *Netatmo Weather Station for Smartphone*. Verfügbar unter: https://www.netatmo.com/de-DE/produkt/wetterstation (01.11.2015).
[72] *The WunderBar is Launched! - relayr blog*. Verfügbar unter: https://blog.relayr.io/the-wunderbar-is-launched/ (14.11.2015).
[73] *Wunderbar | relayr*. Verfügbar unter: https://www.relayr.io/wunderbar/ (01.11.2015).
[74] Relayr, *relayr*. Verfügbar unter: https://developer.relayr.io/dashboard/devices (18.12.2015).
[75] *Glossary*. Verfügbar unter: http://www.support4pc.ch/TippsTricks/TippsTricks.htm (01.11.2015).
[76] Eben Upton, *Raspberry Pi 2 on sale now at $35: Raspberry Pi 2 is now on sale for $35 (the same price as the existing Model B+)*. Verfügbar unter: https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/. (16.01.2015)
[77] M. Zahn, *Unix-Netzwerkprogrammierung mit Threads, Sockets und SSL,* 1. Aufl.: Springer-Verlag, 2006.

Eidesstaatliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Offenburg, den _____

Christian Schneider